

DOS EXTRA

18,- DM 145 Ös 18,- sfr

Sonderausgabe DOS-EXTRA

Nr. 3 1988

DMV-Verlag

Eine Publikation der
DOS-INTERNATIONAL

Der PC – glasklar
Der große Kurs
aus der DOS

DFÜ – Programm
zum Abtippen

Hercules – Grafik
selbstprogrammiert

Sprache digitalisiert
mit dem PC

Seitenweise Tips & Tricks
für MS-DOS-Computer

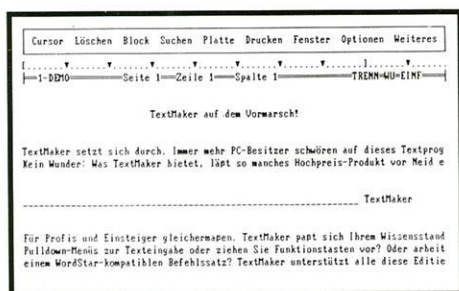


TextMaker auf dem Vormarsch!

TextMaker setzt sich durch. Immer mehr PC-Besitzer schwören auf dieses Textprogramm. Kein Wunder: Was TextMaker bietet, läßt so manches Hochpreis-Produkt vor Neid erblassen.

TextMaker

Für Profis und Einsteiger gleichermaßen. TextMaker paßt sich Ihrem Wissensstand an. Bevorzugen Sie Pulldown-Menüs zur Texteingabe oder ziehen Sie Funktionstasten vor? Oder arbeiten Sie lieber mit einem WordStar-kompatiblen Befehlssatz? TextMaker unterstützt alle diese Editiermethoden.



Neu! Version 1.1

Jetzt auch Unterstützung der Hercules-Karte.
Ab sofort Steuerung auch mit der Maus und neue Befehle.
Auch auf 3,5"-Disketten lieferbar.

Einige der Vorzüge von TextMaker:

- Neun Textfenster
- Textgröße nur durch RAM begrenzt
- Integrierter Druckerspooier
- 26 Tastaturnakros für Texte und Befehle
- Hilfstexte auf Knopfdruck
- Rechnen im Text mit 26 Variablen
- Adressenverwaltung und Serienbrieffunktion
- Phonetisches Suchen: Mayer? Meier? Maier?
- Die UNDO-Taste rettet bis zu 32000 Zeilen
- Layout auf dem Bildschirm zeigen
- Drucken auf Bildschirm: *kursiv*, **fett**, unterstrichen ...
- Frei wählbarer Zeilenabstand
- Integrierter Dateimanager
- Zwischendurch DOS-Befehle eingeben
- Einblendbare ASCII-Tabelle
- Stichwortverzeichnisse, Inhaltsverzeichnisse und Fußnoten
- Automatische Textsicherung in beliebigen Zeitabständen
- Viele Druckertreiber
- Für IBM-PC und Kompatible mit mindestens 256 KByte RAM
- Farbgrafikkarte, Monochrom, Hercules und EGA-Autoswitch

TextMaker erhalten Sie direkt bei SoftMaker und zum Beispiel bei folgenden Fachhändlern:

FAMOS Computer GmbH
An der Kleimannsbrücke 50
4400 Münster
☎ (0251) 322 97

EDV Know-how Bernd Gerdes
Postfach 1227
2165 Harsefeld
☎ (041 64) 58 58

Computer-Vögel
Professor-Prenn-Straße 1
A-6330 Kufstein
☎ (06091) 85 17 32

Unverbindliche Preisempfehlung DM 148,- bei Bezug über Ihren Händler. Bei Versand DM 158,- inkl. Porto und Verpackung.
Lieferung auf Verrechnungsscheck oder per Nachnahme.

nur DM **148,-**

SoftMaker

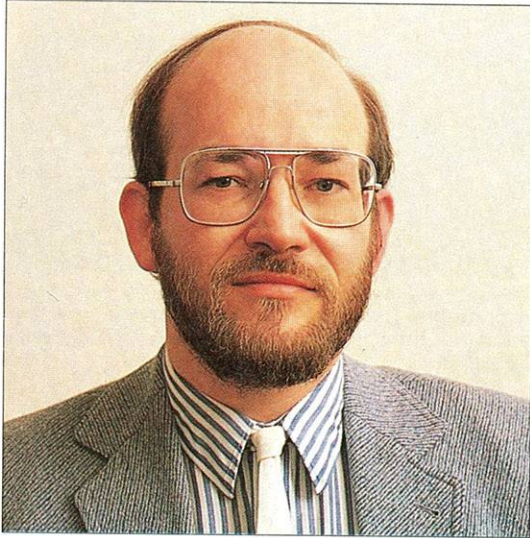
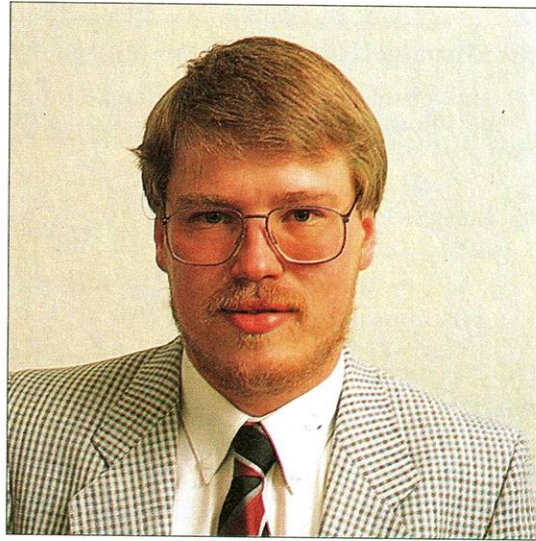
Grabbestraße 9
8500 Nürnberg 90
Telefon (09 11) 30 33 33

Anfragen von Händlern
stets willkommen!

Bitte Coupon ausfüllen, ausschneiden und an SoftMaker schicken.

- | | |
|---|---------------|
| ☐ Bitte schicken Sie mir erst einmal kostenlos Informationen. | Anschrift: |
| ☐ Senden Sie mir eine Demodiskette. Einen 10-Mark-Schein lege ich bei. | |
| ☐ TextMaker ist genau, was ich suche! Schicken Sie mir ein Exemplar! | Unterschrift: |

Was Sie schon immer über Ihren PC wissen wollten



Informationen über seinen PC braucht jeder, egal ob Anwender, Programmierer oder Einsteiger. DOS International bietet Ihnen diese Informationen Monat für Monat in aktueller Form. Tips und Tricks, Utilities, Werkstattberichte – das sind die Themen der DOS International, und unsere neue Schwesterzeitschrift »Window« liefert Ihnen zusätzliche Informationen aus allen Bereichen der Anwendersoftware. Mit diesem dritten DOS Extra halten Sie nun die ideale Ergänzung zu beiden Zeitschriften, zur DOS International und zur neuen Window, in Ihren Händen.

Wie bereits die ersten beiden Extra-Ausgaben der DOS International ist auch das vorliegende dritte Extra einem einzigen, übergreifenden Thema gewidmet. Sie finden hier grundlegende, umfassende Informationen zu den Themengebieten MS-DOS, BIOS, Grafik und Sound. Ganz in der Tradition der DOS International ist das Ganze gewürzt mit interessanten Utilities und natürlich wieder jeder Menge Tips und Tricks.

Kernstück des MS-DOS-Teils in diesem Sonderheft ist der mittlerweile schon legendäre Kurs »Der PC – glasklar«, der letztes Jahr mit großem Erfolg in der DOS International lief. Viele Leserzeitschriften appellierten wegen dieses Kurses schon lange an uns, die schnell vergriffenen ersten DOS-Ausgaben doch nachzudrucken. Nun, der Bitte zum Nachdruck der ersten DOS-Ausgaben können wir aus verschiedenen Gründen leider nicht nachkommen. So haben wir uns aufgrund der großen Nachfrage dazu entschlossen, den PC-Kurs in gründlich überarbeiteter, aktualisierter und erweiterter Form neu zu veröffentlichen und somit auch jenen Anwendern zugänglich zu machen, die nicht zu den DOS-Lesern der ersten Stunde gehören.

Doch der PC-Kurs ist erst der Anfang. Ein weiteres Kapitel in diesem DOS Extra beschäftigt sich mit MS-DOS in allen Schwierigkeitsstufen. Sie erfahren, wie Datenträger aufgebaut sind, und wie Sie einen Treiber für Ihre AT-Tastatur programmieren.

Daneben finden Sie weitere Grundlagen für das Arbeiten mit Ihrem PC. Der Grafikteil bietet Informationen für Pascal- und C-Programmierer.

Eine komplette Grafik-Bibliothek für die Hercules-Karte macht es Ihnen leicht, die großen grafischen Fähigkeiten dieser Karte in Ihren eigenen Pascal-Programmen zu nutzen. Für Turbo C bieten wir Ihnen darüber hinaus eine Bibliothek an Grafikfunktionen, die sich leicht in eigene C-Programme integrieren lassen.

Ein oft unterschätztes Kapitel beim PC ist die Sound-Programmierung. Wir zeigen Ihnen, wie Sie mit der speziell auf diesem Gebiet recht einfachen PC-Hardware dennoch interessante Sound-Effekte in Basic oder Pascal selbst programmieren können. Und hätten Sie gedacht, daß Sie mit Ihrem PC mit wenig Aufwand, ohne teures Zubehör, selbst Sprache digitalisieren und wiedergeben können? Überzeugen Sie sich selbst, DOS Extra 3 informiert Sie darüber, wie Sie Ihrem PC das Sprechen beibringen.

Sie sehen, auch dieses dritten DOS Extra bringt wieder eine hochbrisante Mischung aus tiefgehenden Informationen, verständlicher Darstellung und interessanten Tips und Tricks. Hier finden Sie eben ganz genau das, was Sie schon immer über Ihren PC wissen wollten. Wir sind sicher, daß auch das dritte DOS Extra einen festen Platz auf Ihrem Schreibtisch, direkt neben Ihrem PC, finden wird.

Volker Everts

Andreas Hagedorn

Tips zur Eingabe:

Geprüfte Eingabe – Checksummer	6
-----------------------------------	---

MS-DOS:

»Autoexec« für Einsteiger	10
Die Tastatur kann mehr, als Sie glauben – Ansi-Treiber	14
MS-DOS-Umgebung und Batch-Variablen	18

Grundlagen:

Interrupts und ihre Handhabung beim PC	22
--	----

MS-DOS intern:

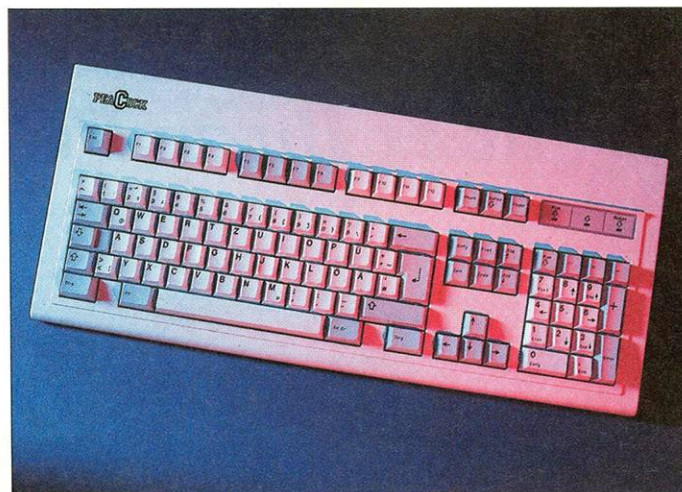
DOS T3 – AT-Tastaturtreiber nach Maß	24
Disketten unter der Lupe – Diskettenaufbau	40
Die Welt der Festplattenspeicher – Festplattenaufbau	42

Kurs:

Der PC – glasklar (Teil 1)	44
Der PC – glasklar (Teil 2)	54
Der PC – glasklar (Teil 3)	64
Der PC – glasklar (Teil 4)	80
Der PC – glasklar (Teil 5)	92

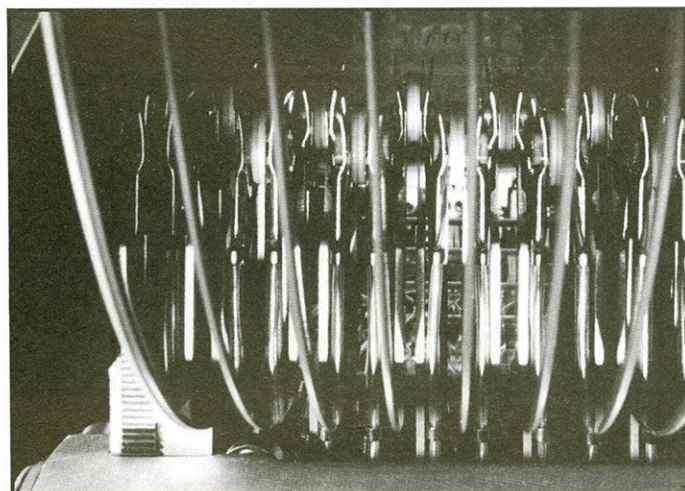
Utilities:

Zeichensatz selbstprogrammiert	100
Datenfernübertragung mit »STrans« – DFÜ-Programm zum Abtippen	118
Wie »STrans« funktioniert	132
Kein Parkverbot – Festplattenköpfe parken	139



Wenn Sie einen AT besitzen und Ihre Tastatur richtig ausreizen oder an Ihre individuellen Vorstellungen anpassen möchten, dann gibt Ihnen der Tastaturtreiber »DOS-T3« dazu die Gelegenheit.

S. 24



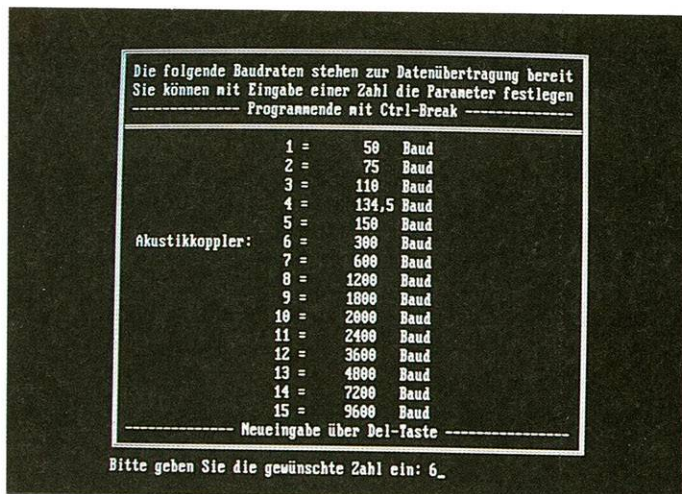
Die Organisation von Diskette und Festplatte ist für viele PC-Besitzer ein Buch mit sieben Siegeln. Doch das Wissen um den Aufbau dieser Datenträger kann in vielen Fällen sehr hilfreich sein.

S. 42



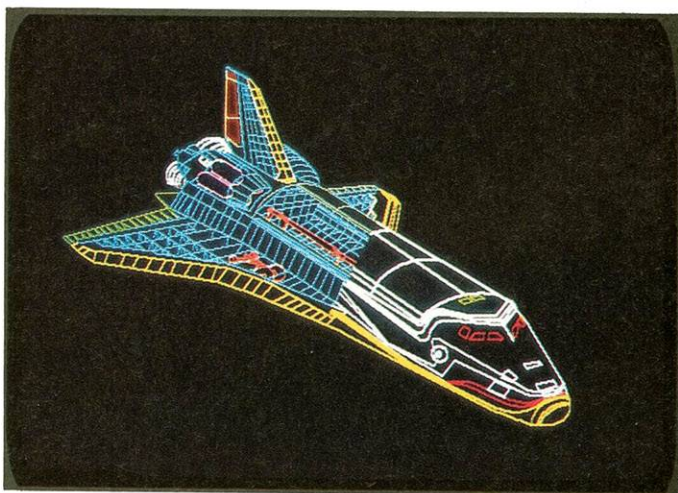
Wenn Sie Ihren PC mehr als nur oberflächlich kennenlernen möchten, dann müssen Sie unbedingt den großen DOS-Kurs »Der PC – glasklar« lesen, der nun in einer neuen, überarbeiteten Form vorliegt.

S. 44



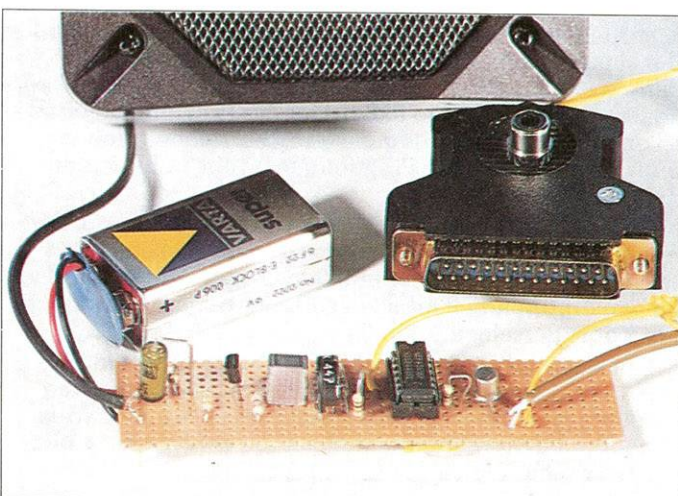
Interessiert Sie das Thema Datenfernübertragung? Das schnelle DFÜ-Programm »STrans« ist zu 100 Prozent in Assembler geschrieben und öffnet Ihnen das Tor zur großen, weiten Datenwelt.

S. 132



Pascal- und C-Programmierer aufgepaßt! Mit der Hercules-Grafik-Toolbox und den C-Grafikfunktionen können Sie Ihre Grafikkarte endlich komfortabel programmieren.

S. 140



Kaum zu glauben, aber wahr: Der PC kann richtig sprechen. Das System »PC-Voice« verwandelt Ihren Computer in eine redselige Maschine und verblüfft damit PC-Besitzer wie – Kritiker gleichermaßen.

S. 166

Tips & Tricks:

Laufwerksinformationen mit GW-Basic	134
Einfacher Datenschutz mit GW-Basic	134
Speichern des Bildschirms mit GW-Basic	134
Die Systemkonfiguration mit GW-Basic ermitteln	134
Laufschriften mit Turbo Pascal	135
Drucker-Unit für Turbo Pascal 4.0	136
Kopieren ohne »copy« mit Turbo Pascal	137
Interrupt-Prozeduren mit Turbo C	138

Grafik:

Die Hercules-Grafik-Toolbox	140
– Grafikfunktionen für Turbo Pascal	
Grafikfunktionen für Turbo C	160

Sound:

Ein PC mit Sound	165
– Grundlagen	
Der PC spricht	166
– Sprachdigitalisierung	
Soundprogrammierung in Basic und Pascal	176

Rubriken:

Editorial	3
Impressum	178
Inserenten	178

Sämtliche

Routinen und Programme

gibt es für 35, – Mark auf

2 Disketten 5 1/4“ oder

1 Diskette 3 1/2“

– auch mit Anpassung für die Hercules-Grafikkarte

Check-summer

Mit SCheck, dem Checksummenprüfprogramm der DOS International, werden alle Listings auf korrekte Eingabe überprüft. Der eingebaute Editor erlaubt sofortige Fehlerkorrektur. Sie sparen damit jede Menge Zeit beim Abtippen von Listings.

Immer mehr Leser der DOS International kaufen sich keine fertige Software, sondern finden in der Zeitschrift Programme abgedruckt, die Ihren Wünschen voll entsprechen.

Damit nun aber die Eingabe möglichst schnell und fehlerfrei erfolgen kann, finden Sie in der DOS sämtliche Listings mit Prüfsumme versehen. Mit Hilfe der Routine SCheck (Listing 1) können Sie Ihre Zeilen mit dem abgedruckten Programm auf Fehler überprüfen. SCheck stellt Ihnen dazu zwei verschiedene Eingabevarianten zur Verfügung.

Beim ersten Weg geben Sie mit dem Editor der entsprechenden Programmiersprache – oder auch mit einem beliebigen Texteditor, der ASCII-Dateien erzeugt – nur den Quellcode ein. Dieser steht hinter der Prüfsumme (in eckigen Klammern) und der Zeilennummer (Zahl mit Doppelpunkt). Listing 2 zeigt eine Pascal-Routine, wie sie im Heft abgedruckt ist, Listing 3 die gleichen Programmzeilen, so wie Sie sie in den Computer eingeben müssen. Eine Programmzeile wird immer durch die Zeilennummer angezeigt und kann dabei im Druck über mehrere Zeilen gehen. In solch einem Fall müssen Sie die Anweisungen durchgängig in einer Zeile in den Editor eintippen.

Als nächstes starten Sie SCheck mit dem Dateinamen

```
scheck <Name.Ext>
```

Legen Sie sich bitte von der eingegebenen Datei eine Kopie an. Zwar arbeitet SCheck nicht mit dem Original, sondern mit einer Kopie, die erst ganz zum Schluß als aktuelle Datei verändert wird, doch wenn ein Fehler beim Speichern auftritt, so ist es nicht ausgeschlossen, daß die mühsam eingetippten Daten verloren gehen. Also immer als erstes eine Kopie der Daten anlegen.

Das Bild zeigt die aktuelle Zeile in dem oberen Fenster. Die Checksumme wird darunter links angezeigt. Diese vergleichen Sie mit der abgedruckten Zahl in den eckigen Klammern. Rechts daneben steht die Zeilennummer und der Name der aktuellen Datei. Im unteren Fenster werden Fehlermeldungen und Hinweise auf die weitere Bearbeitung ausgegeben. Im hier abgebildeten Fall können Sie mit [Return] die Zeile unverändert 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 Cursorstasten steuern Sie den Cursor im Editierfenster. [Home] bringt den Cursor an den Anfang der aktuellen Programmzeile und [End] ans Ende. Löschen können Sie das Zeichen links vom Cursor mit der Taste oberhalb der Enter-Taste (auf der Schreibtastatur). [Del] löscht das Zeichen genau an der Cursorposition. Mit [Return] übernehmen Sie die geänderte Programmzeile. Die Leerzeichen vor der ersten gültigen Anweisung einer Programmzeile werden übrigens bei der Prüfsummenberechnung nicht berücksichtigt. Strukturen können Sie somit nach eigener Wahl einbauen. Weiter hinten stehende Leerzeichen sind für die Prüfsummenberechnung allerdings relevant. Die Tabulatorfunktion dür-

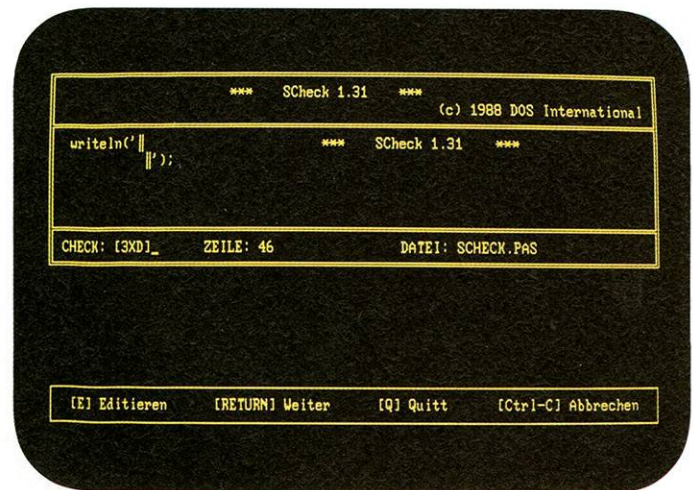


Bild. Mit SCheck können Sie auch falsche Zeilen korrigieren

fen Sie nicht verwenden, wenn Sie mit der Prüfsumme korrekt arbeiten wollen.

Haben Sie so das gesamte Programm durchgearbeitet, so speichert ein letzter Druck auf [Enter] das korrigierte Listing. Das Programm wird beendet und das fehlerfreie Programm steht für die weitere Bearbeitung mit einem Compiler, Assembler oder Interpreter zur Verfügung.

Listing 4 zeigt ein Basic-Programm, wie es in der DOS International abgedruckt wird. Die Eingabe erfolgt wiederum ohne Prüfsumme, aber in diesem Fall mit Zeilennummer. Listing 5 zeigt, wie die Routine eingegeben werden muß.

SCheck erlaubt aber noch einen zweiten Weg, das Listing korrekt einzugeben. Diesmal geben Sie die Prüfsumme mit ein. Starten Sie den Programmierer (wieder gilt, daß die Dateien nur aus ASCII-Werten bestehen dürfen) und geben Sie das Programm mit der Prüfsumme ein. Die eckigen Klammern lassen Sie weg und auch die Zeilennummer. Listing 6 zeigt, wie die Datei mit dem Programm aus Listing 2 aussehen muß. Listing 7 demonstriert das Ganze für das Basic-Programm aus Listing 4.

SCheck starten Sie jetzt mit

```
scheck <Name.Ext> /a
```

Den Schrägstrich »/« können Sie auch durch »-« ersetzen. Desgleichen darf die Option auch als Großbuchstabe angegeben werden. Der Checksummer verwandelt das Programm nun in eine Quelldatei für den entsprechenden Compiler. Dazu werden die ersten vier Zeichen abgeschnitten, die Prüfsumme berechnet und bei einem Fehler die entsprechende Zeile ausgegeben. Der Editor steht dann wieder zur Verfügung. An der Art, wie die Datei verändert wird, sehen Sie, daß hinter der Prüfsumme (ohne Klammern eingegeben) immer ein Leerzeichen stehen muß. Danach folgt der Programmtext (nur in Basic mit Zeilennummer).

Nach dem Durchlauf von SCheck steht wieder das fehlerfreie Programm ohne Prüfsumme auf der Diskette. Und wieder gilt, daß Sie nicht mit dem Original Ihrer stundenlangen Eintipparbeit arbeiten sollten, sondern nur mit einer Kopie derselben.

SCheck verfügt aber noch über eine weitere Option. Die Hexdump-Listings können Sie jetzt nämlich auch ohne den Debugger eingeben. Sie tippen ein Hexdump-Listing wieder mit dem Editor ein und starten dann SCheck ohne Option. Nach

```
scheck <Name.Ext>
```

wird der Text wie im ersten Fall zeilenweise mit Prüfsumme ausgegeben. Der Editor steht wiederum zur Korrektur zur Verfügung.

Befindet sich das Programm korrekt auf der Diskette, so wird SCHECK mit

```
scheck <Name.Ext> /b
```

erneut gestartet. Für die Option gilt das schon oben Gesagte. Auf der Diskette wird nun eine Binärdatei mit der Erweiterung *.bin erzeugt. Für direkt ausführbare Com-Dateien müssen Sie das Programm nur noch mit

```
ren <Name>.bin <Name>.com
```

umbenennen.

Doch auch bei dieser Eingabe werden Sie durch Prüfsummen direkt unterstützt. Geben Sie den Programmtext mit Prüfsumme ein und starten Sie den Checksummer mit

```
scheck <Name.Ext> /a
```

so werden die eingegebenen Daten zeilenweise überprüft. Die so erzeugte Datei entspricht dem Hexdump ohne Prüfsumme.

```
scheck <Name.Ext> /b
```

erzeugt die entsprechende Binärdatei.

Mit SCHECK wird das Eingeben der Programme aus der DOS International bedeutend komfortabler und weniger fehlerträchtig. Wer allerdings kein Turbo Pascal besitzt, um das Programm zu starten, der findet auf der Databox zu der DOS Extra 2 und auf den Databoxen der DOS International ab der Ausgabe 3'88 das Programm als lauffähige Com-Datei. Ihren Listeingaben steht nichts mehr im Wege.

(ev/hg)

```
[BE5] 0: program scheck;
[000] 1:
[BTD] 2: { Ultra-Check-Utility von DOS International
      }
[000] 3:
[ON7] 4: { Volker Everts, 10.1.88 }
[DF5] 5: { Version 1.4 }
[7Z7] 6: { Letzte Änderung: 13.4.88 }
[000] 7:
[000] 8:
[VM1] 9: const
[Y3E] 10: ZEICHEN : string[32] = '0123456789ABCDEF
      HJKLMNPQRSTUWXYZ';
[H42] 11: SPACE = #32;
[RG1] 12: BS = #8;
[DL1] 13: TAB = #9;
[XL1] 14: BEL = #7;
[WP1] 15: CR = #13;
[D32] 16: EMPTY = '';
[RU1] 17: ESC = #27;
[000] 18:
[A21] 19: type
[B67] 20: ZeilenTyp = string[255];
[000] 21:
[EM0] 22: var
[VZ4] 23: Zeile: ZeilenTyp;
[D56] 24: ps, check: string[5];
[26A] 25: Name, TempName, Option: string[40];
[CN4] 26: Ext: string[4];
[GK4] 27: t, temp: text;
[595] 28: f: file of byte;
[5V7] 29: Nummer, b, code: integer;
[9N2] 30: by: byte;
[2D2] 31: ch: char;
[3T7] 32: weiter, Auto, bin: boolean;
[000] 33:
[000] 34:
[BD5] 35: procedure init;
[000] 36:
[EM0] 37: var
[W33] 38: i: integer;
[000] 39:
[BH1] 40: begin
[XV3] 41: Auto := false;
[VX2] 42: bin := false;
[R04] 43: window(1,80,25);
[YES] 44: ClrScr; writeln;
[X4V] 45: writeln('
      ');
[X7E] 46: writeln('*** S
      -Check 1.4
      ');
[Y2H] 47: writeln('
      (c) 1988 DOS Internationa
      ');
[35V] 48: writeln('
      ');
[EAC] 49: writeln('
      ');
[EAC] 50: writeln('
      ');
[EAC] 51: writeln('
      ');
[EAC] 52: writeln('
      ');
[EAC] 53: writeln('
      ');
[35V] 54: writeln('
      ');
[VCE] 55: writeln('CHECK: ZEILE:
      ');
```

```
[V4V] 56: writeln('
      ');
[E03] 57: GotoXY(1,20);
[2SS] 58: writeln('
      ');
[0AC] 59: writeln('
      ');
[EPS] 60: writeln('
      ');
[587] 61: if ParamCount <> 0 then
[H77] 62: for i := 1 to ParamCount do
[BH1] 63: begin
[XE6] 64: Option := ParamStr(i);
[9V3] 65: if (Option = '-a')
[7W3] 66: or (Option = '-A')
[NW3] 67: or (Option = '/a')
[2B9] 68: or (Option = '/A') then Auto := tru
      e
[A11] 69: else
[9V3] 70: if (Option = '-b')
[7W3] 71: or (Option = '-B')
[NW3] 72: or (Option = '/b')
[6H8] 73: or (Option = '/B') then bin := true
[L45] 74: else Name := Option;
[DL0] 75: end
[WC4] 76: else Name := EMPTY;
[X52] 77: repeat
[5N5] 78: if Name = EMPTY then
[BH1] 79: begin
[H03] 80: GotoXY(3,21);
[YKB] 81: write('Geben Sie einen Dateinamen e
      in');
[T93] 82: GotoXY(54,12);
[503] 83: BufLen := 24;
[UM2] 84: read(Name);
[H03] 85: GotoXY(3,21);
[9P5] 86: write('
      ');
[TU0] 87: end;
[BL4] 88: assign(t, Name);
[CB5] 89: {$I-} reset(t); {$I+}
[D25] 90: if IOresult > 0 then
[BH1] 91: begin
[1Y3] 92: for i := 1 to 3 do
[BH1] 93: begin
[M93] 94: GotoXY(10,21);
[E49] 95: write('+++++ DATEIFEHL
      ER +++++', BEL);
[ME2] 96: delay(200);
[M93] 97: GotoXY(10,21);
[086] 98: write('
      ');
[ME2] 99: delay(200);
[TU0] 100: end;
[GE3] 101: Name := EMPTY;
[T93] 102: GotoXY(54,12);
[ZV4] 103: write('
      ');
[TU0] 104: end;
[YM4] 105: until Name <> EMPTY;
[YNF] 106: for i := 1 to length(Name) do Name[i] :=
      UpCase(Name[i]);
[F74] 107: i := pos('.',Name);
[113] 108: if i = 0 then
[BH1] 109: begin
[CT3] 110: Name := Name + '.';
[3P4] 111: i := length(Name);
[TU0] 112: end;
[GF5] 113: Ext := copy(Name, i, 4);
[U66] 114: Name := copy(Name, 1, i-1);
[T93] 115: GotoXY(54,12);
[WT4] 116: write(Name, Ext);
[726] 117: TempName := Name + '.$$$';
[WR6] 118: assign(temp, TempName);
[5L4] 119: rewrite(temp);
[US2] 120: if bin then
[BH1] 121: begin
```



```
[T56] 122: assign(f, Name + '.BIN');
[UA3] 123: rewrite(f);
[XW3] 124: auto := false;
[TU0] 125: end;
[HX2] 126: Nummer := 0;
[TU0] 127: end;
[000] 128:
[000] 129:
[000] 130:
[BZA] 131: procedure Pruefe(var z: ZeilenTyp);
[000] 132:
[06M] 133: { Prüfsumme berechnen. Führende Leerzeichen
und Tabulatoren }
[K4L] 134: { werden nicht berücksichtigt. Bei Steuerzei-
chen im Text }
[DXB] 135: { erfolgt eine Warnung }

[000] 136:
[SC6] 137: var a, p, i, j: integer;
[000] 138:
[BH1] 139: begin
[FD1] 140: p := 0;
[GD1] 141: a := 1;
[9RB] 142: while (z[a] = SPACE) or (z[a] = TAB) do a
:= a + 1;
[BN6] 143: for i := a to length(z) do
[XG4] 144: if z[i] < #32 then
[6Y3] 145: for j := 1 to 3 do
[BH1] 146: begin
[H03] 147: GotoXY(3,21);
[43F] 148: write(' ***** WARNUNG: Steuerzeic-
hen in Zeile ! ', BEL);
[ME2] 149: delay(200);
[H03] 150: GotoXY(3,21);
[AS6] 151: write(' ');
[ME2] 152: delay(200);
[DL0] 153: end
[ZY9] 154: else p := ((i-a+1) and 7) * ord(z[i]) +
p;
[582] 155: ps := ' ';
[MM2] 156: ps[1] := '[';
[AY3] 157: for i := 2 to 4 do
[BH1] 158: begin
[WP7] 159: ps[i] := ZEICHEN[(p and 31) + 1];
[TC2] 160: p := p shr 5;
[TU0] 161: end;
[AN2] 162: ps[5] := ']';
[TU0] 163: end;
[000] 164:
[000] 165:
[46A] 166: procedure edit(var z: ZeilenTyp);
[000] 167:
[EM3] 168: const s = #32;
[000] 169:
[EM0] 170: var
[B43] 171: x: integer;
[2D2] 172: ch: char;
[000] 173:
[598] 174: procedure schreibe(c: char);
[BH1] 175: begin
[D29] 176: if length(z) = 255 then write(BEL)
[A11] 177: else
[BH1] 178: begin
[NA4] 179: insert(c, z, x);
[XY1] 180: x := x + 1;
[SS2] 181: GotoXY(1,1);
[OR2] 182: write(z);
[TU0] 183: end;
[TU0] 184: end;
[000] 185:
[6S5] 186: procedure rechts;
[BH1] 187: begin
[MS8] 188: if x > length(z) then write(BEL)
[DE3] 189: else x := x + 1;
[TU0] 190: end;
[000] 191:
[BS5] 192: procedure links;
[BH1] 193: begin
[U36] 194: if x = 1 then write(BEL)
[RE3] 195: else x := x - 1;
[TU0] 196: end;
[000] 197:
[6A5] 198: procedure hoch;
[BH1] 199: begin
[KY5] 200: if x < 78 then write(BEL)
[WH3] 201: else x := x - 77;
[TU0] 202: end;
[000] 203:
[WU5] 204: procedure runter;
[BH1] 205: begin
[KK9] 206: if x + 77 > length(z) then write(BEL)
[KH3] 207: else x := x + 77;
[TU0] 208: end;
[000] 209:
[2C5] 210: procedure entf;
[BH1] 211: begin
[286] 212: if x <= length(z) then
[BH1] 213: begin
[9T3] 214: delete(z, x, 1);
[SS2] 215: GotoXY(1,1);
[V03] 216: write(z,s);
[DL0] 217: end
[GD4] 218: else write(BEL);
```

```
[TU0] 219: end;
[000] 220:
[MM4] 221: procedure del;
[BH1] 222: begin
[U36] 223: if x = 1 then write(BEL)
[A11] 224: else
[BH1] 225: begin
[4Y3] 226: delete(z, x-1, 1);
[XY1] 227: x := x - 1;
[SS2] 228: GotoXY(1,1);
[V03] 229: write(z,s);
[TU0] 230: end;
[TU0] 231: end;
[000] 232:
[435] 233: procedure pos1;
[BH1] 234: begin
[XD1] 235: x := 1;
[TU0] 236: end;
[000] 237:
[C95] 238: procedure ende;
[BH1] 239: begin
[R64] 240: x := length(z) + 1;
[TU0] 241: end;
[000] 242:
[BH1] 243: begin
[H03] 244: GotoXY(3,21);
[E2T] 245: write(' Editieren Sie die Programmzeile
[RETURN] Ende ');
(**** geändert ****)
[Y14] 246: window(2,6,78,10);
[NF2] 247: ClrScr;
[OR2] 248: write(z);
[SS2] 249: GotoXY(1,1);
[XD1] 250: x := 1;
[X52] 251: repeat
[BU3] 252: read(kbd, ch);
[RS2] 253: case ch of
[WM0] 254: ESC:
[BH1] 255: begin
[BU3] 256: read(kbd, ch);
[RS2] 257: case ch of
[2Y2] 258: 'M': rechts;
[3H2] 259: 'K': links;
[B82] 260: 'H': hoch;
[313] 261: 'P': runter;
[282] 262: 'G': pos1;
[X72] 263: 'O': ende;
[682] 264: 'S': entf;
[TU0] 265: end;
[TU0] 266: end;
[WA2] 267: BS: del;
[WT0] 268: CR: ;
[D35] 269: else schreibe(ch);
[TU0] 270: end;
[PG7] 271: GotoXY(x mod 77, x div 77 + 1);
[4R3] 272: until ch = CR;
[TWE] 273: if auto then z := copy(check, 2, 3) + SPA
CE + z; (**** NEU ****)
[TU0] 274: end;
[000] 275:
[000] 276:
[8V6] 277: procedure ZeigeZeile;
[BH1] 278: begin
[EP3] 279: if Auto then
[BH1] 280: begin
[XF8] 281: check := '[' + copy(Zeile, 1, 3) + '
';
[KSA] 282: check[2] := UpCase(check[2]); (****
NEU ****)
[VSA] 283: check[3] := UpCase(check[3]); (****
NEU ****)
[4TA] 284: check[4] := UpCase(check[4]); (****
NEU ****)
[CU6] 285: Zeile := copy(Zeile, 5, 255);
[TU0] 286: end;
[Y14] 287: window(2,6,78,10);
[NF2] 288: ClrScr;
[OG3] 289: write(Zeile);
[R04] 290: window(1,1,80,25);
[Y93] 291: GotoXY(29,12);
[Y93] 292: write(Nummer);
[TZ3] 293: pruefe(Zeile);
[N93] 294: GotoXY(10,12);
[YR2] 295: write(ps);
[XV4] 296: if bin then exit;
[EP3] 297: if Auto then
[HT4] 298: if ps = check then
[BH1] 299: begin
[H03] 300: GotoXY(3,21);
[CHC] 301: write(' Prüfsumme OK für Zeile ',
Nummer);
[PX4] 302: weiter := true;
[DL0] 303: end
[A11] 304: else
[BH1] 305: begin
[H03] 306: GotoXY(3,21);
[WBS] 307: write(BEL, ' Prüfsummenfehler !
[E] Editieren [RETURN] Weiter [Ctrl-C]
Abbrechen');
[U93] 308: GotoXY(15,12);
[DL0] 309: end
[A11] 310: else
[BH1] 311: begin
[H03] 312: GotoXY(3,21);
```



```
[ZPM] 313: write(' [E] Editieren      [RETURN] W
            eiter      [Q] Quitt      [Ctrl-C] Abbrechen
            ');
[U93] 314:      GotoXY(15,12);
[TU0] 315:      end;
[TU0] 316: end;
[000] 317:
[000] 318:
[XP5] 319: procedure Beenden;
[FK3] 320: var fi: file;
[BH1] 321: begin
[VE5] 322:   while not eof(t) do
[BH1] 323:   begin
[S25] 324:     readln(t, Zeile);
[V36] 325:     writeln(temp, Zeile);
[TU0] 326:   end;
[HY6] 327:   close(t); close(temp);
[CP5] 328:   if bin then close(f);
[PM6] 329:   assign(fi, Name + '.BAK');
[SY5] 330:   {$I-} reset(fi); {$I+}
[HK8] 331:   if Ioresult = 0 then erase(fi);
[D16] 332:   rename(t, Name + '.BAK');
[HT6] 333:   rename(temp, Name + Ext);
[H03] 334:   GotoXY(3,21);
[DD6] 335:   if bin then Ext := '.BIN';
[7HF] 336:   write(' Die Datei ', Name, Ext, ' wurde g
            espeichert ');
[FR7] 337:   while WhereX < 78 do write(' ');
[X03] 338:   GotoXY(1,25);
[5B1] 339:   halt;
[TU0] 340: end;
[000] 341:
[000] 342:
[ES5] 343: procedure Quitt;
[BH1] 344: begin
[H03] 345:   GotoXY(3,21);
[VWK] 346:   write(' Drücken Sie [J], um die Arbeit zu
            beenden ');
[W93] 347:   GotoXY(48,21);
[WE3] 348:   read(kbd,ch);
[3N2] 349:   write(ch);
[3C7] 350:   if (ch = 'j') or (ch = 'J') then
[BH1] 351:   begin
[V36] 352:     writeln(temp, Zeile);
[7U2] 353:     Beenden;
[TU0] 354:   end;
[TU0] 355: end;
[000] 356:
[000] 357:
[BH1] 358: begin
[RB1] 359:   init;
[US2] 360:   if bin then
[BH1] 361:   begin
[H03] 362:     GotoXY(3,21);
[66C] 363:     write(' >>> Binärdatei wird erzeugt <
            <<
            ');
[TU0] 364:   end;
[VE5] 365:   while not eof(t) do
[BH1] 366:   begin
[S25] 367:     readln(t, Zeile);
[X52] 368:     repeat
[N45] 369:       weiter := false;
[U53] 370:       ZeigeZeile;
[867] 371:       if bin then weiter := true;
[Y45] 372:       if not weiter then
[BH1] 373:       begin
[BU3] 374:         read(kbd, ch);
[RS2] 375:         case ch of
[RK4] 376:         CR : weiter := true;
[7R4] 377:         'e','E': Edit(Zeile);
[HK9] 378:         'q','Q': if not Auto then Qui
            tt;
[TU0] 379:       end;
[TU0] 380:       end;
[XY3] 381:       until weiter;
[V36] 382:       writeln(temp, Zeile);
[US2] 383:       if bin then
[BH1] 384:       begin
[WLE] 385:         while copy(Zeile, 1, 1) = SPACE d
            o delete(Zeile, 1, 1);
[X52] 386:         repeat
[RH8] 387:         val('$' + copy(Zeile, 1, 2), b,
            code);
[RY8] 388:         if (code > 0) or (b < 0) or (b
            > 255) then
[BH1] 389:         begin
[H03] 390:           GotoXY(3,21);
[CTD] 391:           write(' +++++ ABRUCH WEGEN
            DATENFEHLER +++++
            ');
[X03] 392:           GotoXY(1,25);
[5B1] 393:           halt;
[TU0] 394:           end;
[VT1] 395:           by := b;
[LH3] 396:           write(f, by);
[OU6] 397:           Zeile := copy(Zeile, 3, 255);
[WLE] 398:           while copy(Zeile, 1, 1) = SPACE
            do delete(Zeile, 1, 1);
[P44] 399:           until Zeile = EMPTY;
[TU0] 400:           end;
[6T5] 401:           Nummer := Nummer + 1;
[TU0] 402:           end;
[7U2] 403:           Beenden;
[5T0] 404: end.
```

Listing 1. SChek prüft jede Zeile auf Fehler

```
[V16] 0:   VERSION = 'Version 2.3';
[H42] 1:   SPACE = #32;
[2G2] 2:   SPACE5 = ' ';
[Y3E] 3:   ZEICHEN : string[32] = '0123456789ABCDEF
            GHLKLMNPRSTUVWXYZ';
[EMO] 4:   var
[F1N] 5:     Nummern, Drucker, Elite, Schmal, Fett, Tx
            t, Pruef, HexDump: boolean;
[FEL] 6:     Summe, Breite, Hoehe, ZeilenNummer, Druck
            Zeile, Zaehler: integer;
[RG2] 7:     p, t: text;
[MN2] 8:     f: BinFile;
[FB2] 9:     LZ: char;
[PM2] 10:    b: byte;
[YB5] 11: procedure info;
[BH1] 12: begin
[9Z2] 13:   writeln;
[EVD] 14:   writeln(BEL, '++++ Fehler bei Parametera
            ngabe !');
```

Listing 2. So werden Pascal-Programme ab sofort gedruckt

```
VERSION = 'Version 2.3';
SPACE = #32;
SPACE5 = ' ';
ZEICHEN : string[32] = '0123456789ABCDEFGLKLMNPRSTUVWXYZ';
var
  Nummern, Drucker, Elite, Schmal, Fett, Txt, Pruef, HexDump: boolean;
  Summe, Breite, Hoehe, ZeilenNummer, DruckZeile, Zaehler: integer;
  p, t: text;
  f: BinFile;
  LZ: char;
  b: byte;
procedure info;
begin
  writeln;
  writeln(BEL, '++++ Fehler bei Parameterangabe !');
```

Listing 3. So müssen Sie das Pascal-Listing eingeben

```
V16 VERSION = 'Version 2.3';
H42 SPACE = #32;
2G2 SPACE5 = ' ';
Y3E ZEICHEN : string[32] = '0123456789ABCDEFGLKLMNPRSTUVWXYZ';
EMO var
F1N Nummern, Drucker, Elite, Schmal, Fett, Txt, Pruef, HexDump: bool
FEL Summe, Breite, Hoehe, ZeilenNummer, DruckZeile, Zaehler: integer
RG2 p, t: text;
MN2 f: BinFile;
FB2 LZ: char;
PM2 b: byte;
YB5 procedure info;
BH1 begin
9Z2 writeln;
EVD writeln(BEL, '++++ Fehler bei Parameterangabe !');
```

Listing 4. Pascal-Programme mit Prüfsumme eingegeben

```
[PRC] 1010 SCREEN 9:KEY OFF:CLS:COLOR 1,0:PALETTE 1,7:P
            ALETTE 7,1
[6U4] 1020 COLORSW=1:GOTO 1100
[213] 1025 RESUME 1030
[H3B] 1030 ON ERROR GOTO 0:SCREEN 2:KEY OFF:CLS:COLORSW
            =0
[164] 1040 VY1=199:VY2=150
[CDK] 1100 LOCATE 10,1:PRINT "Please type the longitude
            of your country"
```

Listing 5. Basic-Programme sehen ab sofort in der DOS so aus

```
1010 SCREEN 9:KEY OFF:CLS:COLOR 1,0:PALETTE 1,7:PALETTE 7,1
1020 COLORSW=1:GOTO 1100
1025 RESUME 1030
1030 ON ERROR GOTO 0:SCREEN 2:KEY OFF:CLS:COLORSW=0
1040 VY1=199:VY2=150
1100 LOCATE 10,1:PRINT "Please type the longitude of your country"
```

Listing 6. Bei der Eingabe müssen Sie auf die Zeilennummer in eckigen Klammern verzichten

```
PRC 1010 SCREEN 9:KEY OFF:CLS:COLOR 1,0:PALETTE 1,7:PALETTE 7,1
6U4 1020 COLORSW=1:GOTO 1100
213 1025 RESUME 1030
H3B 1030 ON ERROR GOTO 0:SCREEN 2:KEY OFF:CLS:COLORSW=0
164 1040 VY1=199:VY2=150
CDK 1100 LOCATE 10,1:PRINT "Please type the longitude of your country"
```

Listing 7. Auch Basic-Programme können mit Prüfsumme eingegeben werden

»Autoexec« für Einsteiger

Eine spezielle Stapelprozedur von MS-DOS wird ausgeführt oder sogar verändert, ohne daß der Benutzer dies zunächst bemerkt. »autoexec.bat« heißt die Datei, die beim Betrieb eines PC eine wichtige Rolle spielt.

Betrachten Sie einmal, was passiert, wenn Sie Ihren PC – entweder durch normales Einschalten oder den berühmten »Affengriff« [Ctrl-Alt-Del] starten. Dieses »Booten«, wie man es nach dem englischen »to bootstrap« (sich selbst die Schnürsenkel zuziehen) nennt, läuft in zwei beziehungsweise drei Phasen ab. Die erste ist die Prüfung der Hardware auf Funktionsfähigkeit. Das BIOS (Basic Input Output System, grundlegendes Ein-/Ausgabesystem) im internen, nicht überschreibbaren Speicher des PC kontrolliert den Bildschirm, die Tastatur, den Hauptspeicher und die restliche Hardware. Das passiert nur beim Einschalten – dem Kaltstart – aber nicht beim Warmstart mit [Ctrl-Alt-Del].

In der folgenden Phase wird das Laufwerk A nach den Startdateien des MS-DOS abgesucht. Dies sind im wesentlichen die beiden Systemdateien »ibmbio.com« und »ibmdos.com«, die im Inhaltsverzeichnis des Datenträgers unsichtbar sind, denn sie haben das Dateiattribut »hidden« (verborgen). Zusätzlich dazu wird der Befehlsprozessor des Betriebssystems »command.com« benötigt, der alle Befehle interpretiert und ihre Ausführung veranlaßt. Findet sie das Betriebssystem in Laufwerk A nicht, oder ist dieses nicht bereit, so wird auf einer eventuell vorhandenen Festplatte C weitergesucht. Im Normalfall werden diese drei Routinen in den Hauptspeicher geladen, und sie übernehmen dann die weitere Steuerung.

Damit beginnt die letzte Phase, die mit der Suche nach der Datei »config.sys« eingeleitet wird, die bestimmte Aspekte der Systemkonfiguration (daher der Name) beschreibt. Nachdem alle ihre Parameter verarbeitet worden sind (dazu kann zum Beispiel das Einrichten einer RAM-Disk im Hauptspeicher gehören), macht sich das System auf die Suche nach einer Stapelprozedur namens »autoexec.bat«. Diese scheinbar so spezielle Stapeldatei unterscheidet sich von anderen und ihren Ausführungsmechanismen ausschließlich dadurch, daß sie von MS-DOS automatisch »exekutiert« wird (was im englischen »ausführen« bedeutet). Niemand hindert Sie daran, an jeder beliebigen Stelle im Betriebssystem den Befehl

```
autoexec
```

einzugeben, um den beim Systemstart gewünschten Zustand wiederherzustellen. Denn genau das ist die Aufgabe dieser Prozedur: Die Herstellung eines gewünschten Zustandes.

Nach dem Starten des Systems sollten zumindest Datum und Uhrzeit korrekt angezeigt und die deutsche Tastaturbelegung mit »keybgr.com« geladen werden. Danach möchten Sie die Version des Betriebssystems am Bildschirm anzeigen lassen, um zu wissen, womit Sie arbeiten und welche Befehle Ihnen zur Verfügung stehen. Also wird Ihre einfachste Version der »autoexec.bat« aussehen wie in Listing 1.

[RD2]	0: keybgr
[P01]	1: date
[161]	2: time
[PM0]	3: ver

Listing 1.
Einfache Startprozedur
»autoexec.bat«

Bevor wir auf weitere und interessante Möglichkeiten der Startprozedur eingehen, wenden Sie gleich einen kleinen, aber wirkungsvollen Trick an. Sie wissen ja, daß Stapelprozeduren Zeile für Zeile von der Diskette oder Platte gelesen und interpretiert werden. Das ist einem ungeduldgigen MS-DOS-Benutzer natürlich zu langsam.

Am Anfang war die Zeit... und die richtige Tastatur.

Wozu haben Sie denn Ihre schnelle »Diskette«, die RAM-Disk im Hauptspeicher eingerichtet? Sie geben Ihrer eigentlichen Startprozedur einen beliebigen Namen (hier heißt sie »autoexec1.bat«), kopieren sie bei Arbeitsbeginn auf die RAM-Disk »c:«, führen sie dann aus und halten damit Ihre Prozedur auf dem langsamen Speicher so klein wie in Listing 2. Das Kopier-Protokoll wird durch den Zusatz

```
>nul
```

unterdrückt und danach die Ausführung der kopierten Prozedur gestartet. Neben der Schnelligkeit kommen Sie noch in den Genuß eines weiteren Vorteils: Ihre »eigentliche« Startprozedur mit dem Namen »autoexe1.bat« (oder wie immer sie auch heißen mag) bleibt erhalten. Eine nicht unbedingt selbstverständliche Angelegenheit, denn viele Programmpakete enthalten sogenannte Installationsroutinen (oft mit dem Namen »install«), mit denen sie den Gegebenheiten des jeweiligen PC angepaßt werden können (etwa durch die Installation des richtigen Druckertreibers). Viele dieser Installationsroutinen verändern auch die »autoexec.bat«, beziehungsweise legen eine völlig neue, an ihre Bedürfnisse angepaßte Datei an. Professionelle Software warnt den Benutzer zuvor, aber es soll ja auch andere geben – und weg ist Ihre so sorgfältig erzeugte Startprozedur.

Nachdem Sie nun das Prinzip der »autoexec.bat« kennengelernt haben, sollen Sie eine praxisnahe Version zusammenstellen, die Sie dann nach Belieben Ihren Bedürfnissen anpassen können. Alles, was Sie als Handwerkszeug dazu brauchen, ist ein normaler Texteditor, der ASCII-Dateien anlegen kann. Je nach Belieben und Gegebenheiten können Sie dabei die Variante mit oder ohne RAM-Disk wählen.

Den Inhalt dieser Prozedur sehen Sie in Listing 3. Zuerst wird in einer »Umgebungsvariablen« mit einem frei gewählten Namen (hier »LW« für »Laufwerk«) das RAM-Laufwerk gespeichert, um es später darüber anzusprechen und die Zuweisung somit flexibler zu halten (bei einem PC mit Festplatte wäre das RAM-Laufwerk D). Diese Variable der MS-DOS-Umgebung wird dann bei den folgenden Befehlen verwendet, denn nun kopieren Sie den Kommando-Prozessor und die wichtigsten externen MS-DOS-Befehle (wie »label«, »find«, »sort« und so weiter) auf die »schnelle Diskette« und teilen dem Betriebssystem durch

```
set comspec=c:\command.com
```

mit, daß es den Prozessor gegebenenfalls von ihr aus nachladen soll. Zusätzlich kopieren Sie nach Belieben weitere Dateien, zum Beispiel »cr« (siehe unten). Soweit die Zeilen 06 bis 16 der Prozedur.

Zuvor mußten Sie noch die Tastaturanpassung vornehmen und ein speicherresidentes Utility-Paket laden (Zeile 04 und 05). Hier nehmen Sie natürlich Variationen Ihren Wünsche gemäß vor.

Die erwähnte Hilfsdatei »cr« benötigt man häufig zur »Eingabeumleitung«, etwa in der Form

```
date <cr
```

Hierbei wird die Ausgabe des Befehls »date« angezeigt, jedoch die Benutzereingabe unterdrückt, beziehungsweise aus der Datei »cr« geholt, die im allgemeinen leer ist. Das


```
[1Y6] 0: copy autoexec1.bat c: >nul
[P13] 1: c:autoexec1
```

Listing 2. Angepaßte Startprozedur »autoexec.bat«

```
[3H2] 0: echo off
[696] 1: rem RAM-Disk Laufwerk:
[4F2] 2: set lw=c:
[R62] 3: keybgr
[WC3] 4: utility
[RZ8] 5: set comspec=%lw%\command.com
[W57] 6: copy command.com %lw% >nul
[9H7] 7: copy diskcopy.com %lw% >nul
[717] 8: copy more.com %lw% >nul
[W27] 9: copy sort.exe %lw% >nul
[P37] 10: copy label.com %lw% >nul
[MY6] 11: copy find.exe %lw% >nul
[167] 12: copy sichern.bat %lw% >nul
[8C7] 13: copy dienst.bat %lw% >nul
[XR4] 14: copy cr %lw% >nul
[9B7] 15: copy typeout.bat %lw% >nul
[VK2] 16: echo on
[BN5] 17: prompt $e[0;65;"CLS";13p
[WN5] 18: prompt $e[0;66;"CD\";13p
[NA7] 19: prompt $e[0;67;"DIR *.bat";13p
[DNG] 20: prompt $e[s$e[1;1f$e[K$e[37;44m F7: cls F8:
    cd\ F9: dir *.bat
[3SA] 21: $e[35;40m$a Datum: $d $e[u$e[37
    ;40m$P$g$e[32;40m
[WR3] 22: path %lw%\a:\
[MM0] 23: cls
[3M0] 24: rem
[4EH] 25: rem command.com auf %lw%, ^U = Pop-UP-Utili
    ties (Uhr, Text, ...)
```

Listing 3. Zusätzliche Startprozedur »autoexe1.bat«

weitere Dateien, zum Beispiel »cr« (siehe unten). Soweit die Zeilen 06 bis 16 der Prozedur. Zuvor mußten Sie noch die Tastaturanpassung vornehmen und ein speicherresidentes Utility-Paket laden (Zeile 04 und 05). Hier nehmen Sie natürlich Variationen Ihren Wünschen gemäß vor.

Die erwähnte Hilfsdatei »cr« benötigt man häufig zur »Eingabeumleitung«, etwa in der Form

```
date <cr
```

Hierbei wird die Ausgabe des Befehls »date« angezeigt, jedoch die Benutzereingabe unterdrückt, beziehungsweise aus der Datei »cr« geholt, die im allgemeinen leer ist. Das heißt, sie enthält nur den Wagenrücklauf (»carriage return«). Sie läßt sich leicht selbst herstellen:

```
A>copy con cr [return] [ctrl-Z]
```

Nun haben Sie die Prozedur bis Zeile 16 eingerichtet. Ungeduldige können jetzt schon einmal die neue Diskette zum »Booten« verwenden, also das MS-DOS mit der Tastenkombination [Ctrl-Alt-Del] starten, um die Korrektheit der bisherigen Schritte zu überprüfen, zum Beispiel durch

```
dir c:
```

Wenn Sie aber Ihrer Sache sicher sind, nehmen Sie doch gleich die Tastaturerweiterung vor, wie in »autoexe1.bat« ab Zeile 18 gezeigt. Hier nur das Wichtigste in Stichworten: Der mit der Konfigurationsdatei installierte Tastaturreiber »ansi.sys« soll mit Escape-Sequenzen versorgt werden, um verschiedene nützliche oder angenehme Einstellungen vorzunehmen. Sie erzeugen die entsprechenden Codesequenzen mit dem Prompt-Befehl, dessen Ausgabe aber am Bildschirm angezeigt werden muß, um den Tastaturreiber zu aktivieren. Also schalten Sie in Zeile 17 die Bildschirmanzeige ein und erzeugen in Zeile 18 die Sequenz aus Esc-Zeichen und »eckiger Klammer auf« ([), gefolgt von der Zeichenkette »0;65;"CLS";13p«. Durch diesen »Befehl« belegt der Ansi-Treiber die Funktionstaste [F7] (die den Tastaturcode 65 hat) mit dem MS-DOS-Befehl »cls« zum Löschen des Bildschirms. In gleicher Weise

wird danach auf [F8] die Rückkehr in das Hauptverzeichnis gelegt und auf [F9] die Anzeige aller Bat-Dateien. Nun folgt ein wahres Prompt-Ungetüm, das in diesem Artikel auch bereits ausführlich erläutert wurde: Die Anzeige der gerade definierten Funktionstasten am Bildschirmkopf zur ständigen Information des Benutzers, die Anzeige des Datums (die Uhrzeit kann in gleicher Weise mit der Prompt-Zeichenkette »\$t« angezeigt werden, es wurde hier aber die Anzeige von »utility« verwendet) und schließlich die Anzeige des gerade gültigen Verzeichnisses (durch die Prompt-Sequenz »\$p\$g«). Schließlich (Zeile 22) wird noch der Zugriffspfad selbst angegeben: Das Durchsuchen der RAM-Disk und des Laufwerkes A nach ausführbaren Dateien. Dann wird noch der Bildschirm gelöscht und mit einem Kommentar auf die residenten Utilities hingewiesen. Nun steht das System dem Benutzer zur Verfügung.

Natürlich könnte man noch mehr Komfort anbieten, etwa eine eigene »Menüsteuerung«. In Anführungsstrichen steht diese Menüsteuerung deshalb, weil es keine echte ist, sondern nur eine solche simuliert. Schreiben Sie in »autoexec1.bat« statt Zeile 25 einfach

```
hilfe
```

(das ist nichts anderes als der Aufruf einer weiteren Stapelprozedur namens »hilfe.bat«), und schreiben Sie in diese hinein

```
type hilfe
```

»hilfe« ist dann eine einfache Textdatei nach dem Muster im gezeigten Listing, die Sie mit Hilfe des Texteditors leicht herstellen können. Hinter »1«, »2« etc. verbergen sich kleine Batch-Dateien, etwa in der Form »1.bat«

```
echo Bitte Multiplan-Diskette in Laufwerk B: einlegen! pause
```

```
b:
```

```
cd \mp
```

```
mp
```

```
a:
```

```
type hilfe
```

Im Laufwerk B können Sie also mit Anwendungsprogramm-Disketten arbeiten und nach Arbeitsende wieder Ihr Menü anzeigen lassen. So kommen Sie auch ohne die Betriebssystem-Diskette aus. Denn viele MS-DOS-Befehle sind ja resident, stehen also immer zur Verfügung, und andere halten Sie ja auf Ihrer RAM-Disk bereit.

Fassen wir die wesentlichen Tätigkeiten Ihrer Startprozedur (ohne die zusätzliche Menüsteuerung; vergleichen Sie Listing 3) noch einmal zusammen: Sie verlagert den MS-DOS-Kommandoprozessor auf die RAM-Disk, ebenso einige häufig gebrauchte, externe MS-DOS-Befehle; sie belegt einige Funktionstasten mit speziellen, von Ihnen gewünschten Befehlen und zeigt diese ständig am oberen Bildschirmrand an; sie definiert schließlich als Suchpfad für Befehle und Programme zuerst die RAM-Disk, dann das Standardlaufwerk A.

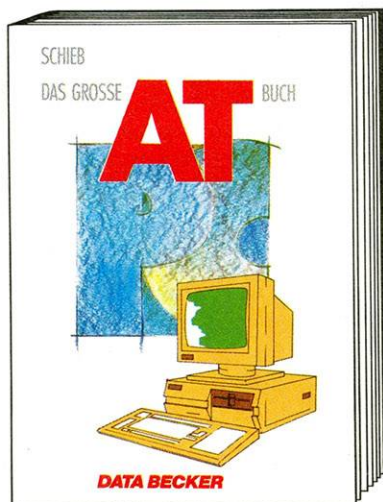
Mit diesem Vorgehen haben Sie nicht nur Ihr Verständnis des MS-DOS geübt und vertieft, sondern außerdem einen ständigen Nutzeffekt für Ihre weiteren Arbeiten geschaffen.

(Jürgen Beetz/fl/dm)

Datum ist: So. 20.03.1988 Zeit ist: 15.12.10.32			
3 Unterverzeichnisse (s. Liste), RAM-Disk "C:" mit 64 kB weitere Informationen mit HILFE MENÜ.			
Auf der DOS-Anweisungszeile ('A\>') die Nummer des Programmes eingeben, mit dem gearbeitet werden soll:			
Eingabe	Programm	Directory	Anmerkungen
HILFE			
1	Multiplan	MP	Dieses Menü
2	Wordstar2000	WS2	Version 1.0) Disketten-
3	dBase II	DB2	Version 1.2) wechsel in
			Version 1.0) Laufwerk B:
U	Utilities		(Untermenü)

Listing 4. Hilfe- beziehungsweise Auswahlanzeige

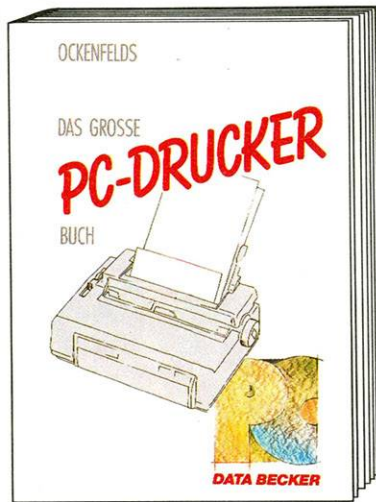
NEUE PC



Den PC/AT verstehen.

In diesem Buch erfahren Sie alles, was Sie zum AT wissen wollen. Die Entwicklung, die Gemeinsamkeiten und Unterschiede zum Vorgänger PC/XT, das Installieren des Systems, die Verwendung der Betriebssysteme DOS und OS/2, die Programmierung der Tastatur, die Organisation der Festplatte, die Unterschiede der Videokarten (MDA, CGA, EGA, 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,-



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 zu allen Druckerproblemen und -fragen. Ganz neu: mit beiliegender Diskette.

Das große PC-Drucker-Buch
Hardcover, inkl. Diskette, ca. 390 Seiten, DM 59,-
erscheint ca. 5/88



OS/2, das Betriebssystem von morgen

Die gesamte Fachpresse spricht von OS/2, dem Betriebssystem der Zukunft. Das große Buch zu OS/2 hilft Ihnen sowohl beim Einstieg als auch bei der professionellen Nutzung. Aus dem Inhalt: die Benutzeroberfläche, das Installieren, die Befehle, das Erstellen von Batchprogrammen, das Konfigurieren des Systems durch die Datei CONFIG.SYS. Die Programmierer und weitergehend Interessierten erfahren alles über Threads, Prozesse, Semaphore, Prioritäten, die Programmierung unter OS/2, Multitasking und vieles andere mehr. Kurz – in diesem Buch finden Sie alle Informationen, die Sie für die Arbeit mit OS/2 benötigen.

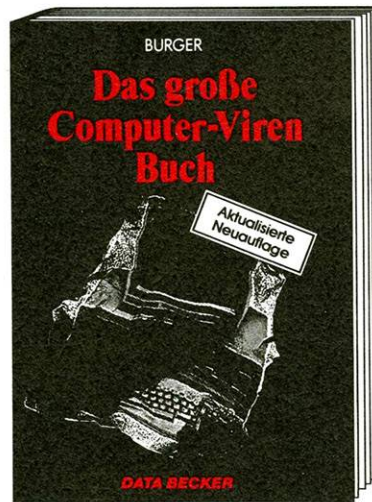
Das große OS/2-Buch
Hardcover, 460 Seiten, DM 49,-



Turbo Pascal 4.0 für Einsteiger

Turbo Pascal 4.0 für Einsteiger – da macht der Einstieg Spaß. Leichtverständlich und behutsam führt Sie hier ein Profi an die Programmierertechnik unter Turbo Pascal 4.0 heran. Typische Einsteigerfehler werden aufgezeigt, und so beherrschen Sie schon nach kürzester Zeit alle wichtigen Grundlagen dieser neuen Programmiersprache und sind in der Lage, Ihre ersten lauffähigen Turbo-Pascal-Programme zu schreiben. Einen besseren Einstieg kann man kaum finden.

Turbo Pascal 4.0 für Einsteiger
244 Seiten, DM 29,-



Die aktualisierte Neuauflage zu einem heißen Thema.

Kaum einer hat sich so intensiv mit dem Thema Computer-Viren beschäftigt wie Ralf Burger. In diesem Buch gibt er sein ganzes Wissen weiter. Detailliert und umfassend beschreibt er die faszinierend schreckliche Welt der Computer-Viren. Mit Programm listings von Demonstrations-Virenprogrammen und vielen Beispielen zu möglichen Manipulationen durch Viren. Dazu zahlreiche Erfahrungsberichte von Hackern und Managern. Ein Buch, bei dem Ihnen nichts verborgen bleibt.

Das große Computer-Viren-Buch
363 Seiten, DM 49,-



Die DATA BECKER Führer – immer aktuell:

Hier drei weitere Neuerscheinungen der erfolgreichsten und komplettesten Serie ihrer Art. Programmwissen pur im Westentaschenformat. Kompetent, schnell und immer griffbereit.

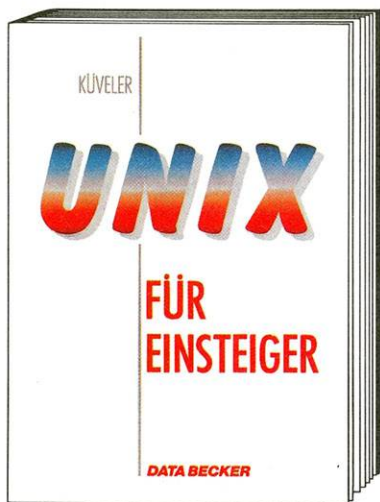
DATA-BECKER Führer
Word Perfect
inkl. Version 4.2 deutsch
191 Seiten, DM 29,80

DATA BECKER Führer
Turbo Pascal
inkl. Version 4.0 deutsch
126 Seiten, DM 24,80

DATA BECKER Führer
Word
inkl. Version 4.0 deutsch
176 Seiten, DM 29,80

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 31 00 10

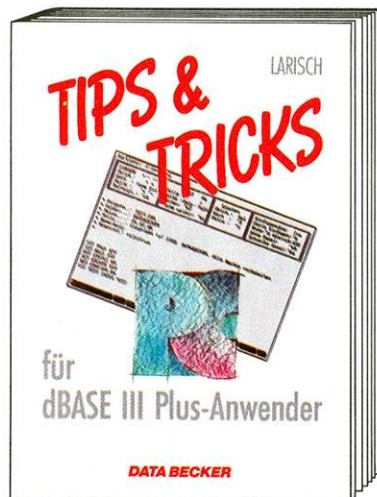
BÜCHER



UNIX für Einsteiger

Ihre ersten Schritte in die UNIX-Welt sollten Sie mit diesem Buch beginnen. Leichtverständlich erhalten Sie das Grundlagenwissen, das Sie für die Arbeit mit UNIX oder einer seiner Varianten benötigen. Aus dem Inhalt: Geschichte und Besonderheiten, Unterschiede zu anderen Betriebssystemen wie MS-DOS, CP/M und OS/2, UNIX-Kommandos, die Shell, Dateiverwaltung, Rechnersteuerung mit Filtern, Pipes, Wildcards und Hintergrundprozessen. Weiterhin finden Sie Übungen zur Arbeit mit UNIX, Erläuterung der Editoren und Hinweise zur Programmiersprache C, zum Superuser, zur Textverarbeitung und Programmentwicklung.

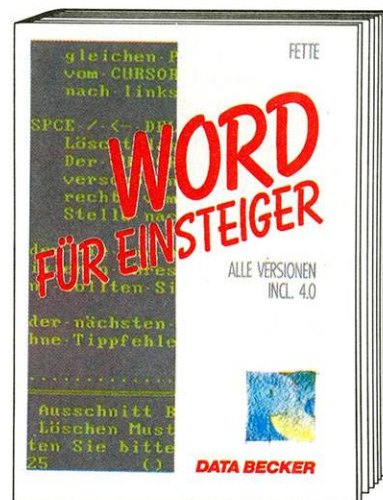
UNIX für Einsteiger
215 Seiten, DM 49,-



dBase III Plus - Tips & Tricks

Allen Anwendern, die bereits mit dBase III Plus arbeiten, gibt dieses Buch den letzten Schliff. Hier finden Sie alles, was die Arbeit mit diesem ausgefeilten Datenbank-System richtig professionell gestaltet: Merkmale der Punktoberfläche, Variablenbehandlung, Datenverwaltung, Programloptimierung, Daten- und Programmübernahme aus früheren Versionen, Datenaustausch und als besonderes Bonbon einen kleinen Ausblick auf das, was das neueste dBase IV von Ashton Tate zu bieten hat. Dazu jede Menge Hilfstoutinen und zahlreiche Tips zur praktischen Arbeit. Besonders hilfreich: die Diskette zum Buch mit allen Beispielen.

Tips & Tricks für dBase III Plus-Anwender
Hardcover, 444 Seiten, inkl. Diskette, DM 69,-



Word für Einsteiger

Word starten und gleich die ersten Texte schreiben. Mit Word für Einsteiger. Bereits nach einem Abend beherrschen Sie die wichtigsten Grundlagen dieser komplexen Textverarbeitung. Textformatierung, Serienbriefe und das Ausdrucken bleiben keine Geheimnisse mehr. Neben den Word-Versionen 2, 3 und Junior werden auch die erheblichen Erweiterungen der Word-Version 4 erklärt: Datei-Manager zur Dokumentenverwaltung, neue Funktionen zur Textüberarbeitung und die Menüführung.

Word für Einsteiger
Hardcover, 267 Seiten, DM 49,-



Wie gedruckt. Mit WORD 4.0!

Microsoft WORD bietet alles, was man von einer Textverarbeitung erwarten kann. „Zuviele“, meint so mancher Anwender. Schließlich ist es nicht ganz einfach, die zahlreichen Optionen und Funktionen dieses Textprogramms auch komplett zu beherrschen. Nun ist mit der neuen Version 4.0 von WORD die bewährte Version 3.0 abermals erheblich erweitert worden. Und das sicherlich nicht, um den Anwender noch mehr zu verwirren. Grafikeinbindung, Makro-Funktionen und zahlreiche Detailverbesserungen machen jetzt eine ansprechende Gestaltung von Schriftstücken geradezu zum Kinderspiel – vorausgesetzt, man verfügt über das nötige Know-how:

„Wie gedruckt – DTP mit WORD 4.0“. Hier finden Sie alles für wirklich anspruchsvolle WORD-Anwendungen: Erstellen von Druckformaten für Briefe, längere Texte, Tabellen und Formulare, Serienbriefe mit WORD, Arbeiten mit Makros, Übernahme von Daten aus Kalkulationsprogrammen, Fußnoten mit WORD, Grafiken mit WORD, und und und. Natürlich mit zahlreichen Beispielen und vielen Tips aus der Praxis eines WORD-Profis. Eben ein Buch für alle, die mehr über WORD aber auch über Desktop-Publishing wissen wollen. Aus der Kombination von beidem ergeben sich ganz andere Perspektiven. Dieses Buch beweist es.

Straßburg
Wie gedruckt – Desktop Publishing mit Word 4.0
Hardcover, ca. 250 Seiten, DM 79,-
erscheint ca. 5/88

COUPON

COUPON BITTE EINSENDEN AN:
DATA BECKER
MEROWINGERSTR. 30
4000 DÜSSELDORF

HIERMIT BESTELLE ICH

NAME, VORNAME

STRASSE

ORT

Die Tastatur kann mehr, als Sie glauben

Im MS-DOS werden angeschlossene Ein-/Ausgabe-Einheiten (zum Beispiel Tastatur, Bildschirm, Drucker) durch vorhandene Routinen, die »Einheitentreiber«, angesteuert. Die vorhandenen Treiber können Sie aber auch durch eigene ersetzen. Was zuerst wie eine Spielerei für Spezialisten aussieht, entpuppt sich als eine höchst nützliche Einrichtung.

»Einheitentreiber« – ein zumindest für den Anfänger ungewohnter Begriff – bedeutet, daß Ein-/Ausgabe-Einheiten nicht durch Hardware, sondern durch Programme betrieben werden. Zwei solcher Programme werden mit dem MS-DOS mitgeliefert, sind aber standardmäßig nicht in Betrieb, sondern müssen extra installiert werden. Nichts leichter als das: Das Betriebssystem schaut beim Anlaufen in einer speziellen Datei namens »config.sys« nach, um seine System-Konfiguration zu erfahren. Installieren Sie selbst eine weitere Einheit, nämlich eine RAM-Disk, also eine virtuelle Platte, die Sie wie ein Laufwerk ansprechen können. Sie ist jedoch in Wirklichkeit im Hauptspeicher vorhanden und daher sehr schnell. Eine solche Einheit wird in der Konfigurationsdatei mit dem »Device-Befehl« angegeben, in dem der gewünschte Einheitentreiber genannt wird. Im Fall der mit dem MS-DOS gelieferten virtuellen Platte ist es »vdisk.sys«. Nehmen Sie also einen beliebigen Zeilen-Editor und fügen Sie in Ihrer Konfigurationsdatei hinzu:

```
device=vdisk.sys 64
```

Und schon haben Sie eine virtuelle Platte von 64 KByte Größe (oder was immer Sie hinter »vdisk.sys« angegeben haben). Allerdings wird diese Änderung nicht sofort wirksam, denn die Eintragungen in der Konfigurationsdatei werden nur beim Start von MS-DOS berücksichtigt. Also müssen Sie mit dem »Warmstart« durch die Tastenkombination [Ctrl-Alt-Del] das Betriebssystem noch einmal in Gang setzen. Nun kopieren Sie eine Datei auf Ihre virtuelle Platte (»C« für PCs mit 2 Disketten, »D« für Diskette und Festplatte) und überzeugen sich mit dem entsprechenden Dir-Befehl, daß sie auch wirklich angekommen ist.

Eine virtuelle Platte von genügender Größe ist für viele Anwendungen nützlich, um deren Ablauf zu beschleunigen. Als Einsatzmöglichkeit einer RAM-Disk bietet sich an, häufig benötigte externe MS-DOS-Befehle wie »find« oder »sort« dorthin zu verlagern oder den Befehlsprozessor darin zu speichern, sie also als »Systemdiskette« zu verwenden. Durch Befehle auf der RAM-Disk erhalten Sie einen nicht unbedeutenden Geschwindigkeitsvorteil, besonders wenn Sie bedenken, daß bei der Datenübergabe vom MS-DOS automatisch Zwischendateien angelegt werden. Ein Kommando wie

```
dir | sort /+14
```

mit dem Sie die Ausgabe des Dir-Befehls nach der Dateigröße sortieren, wird auf der RAM-Disk spürbar schneller. Weiterhin profitieren davon Programme, die deswegen etwas langsam und gemütlich ablaufen, weil sie alle Augenblicke Programmteile nachladen. Diese »Overlays« (oft erkennbar am Zusatz »_ovr« auf der Diskette) überlagern im Hauptspeicher schon vorhandene Programmteile und verhindern so, daß das Programm dort zuviel Platz einnimmt. Speichert man diese also auf der virtuellen Platte, geht das Nachladen blitzschnell ohne mechanische Bewegungen vor sich.

Ein Problem sollte aber nicht verschwiegen werden: Die »vdisk« liegt ja gerade im Hauptspeicher, wird damit von ihm abgezogen, verbraucht also genau den Platz, den man mit den Overlays einsparen wollte. Mit anderen Worten: unter mindestens 512 KByte Hauptspeicher ist eine RAM-Disk nicht sinnvoll. Aber die Speicherchips sind ja billig und viele PCs haben schon standardmäßig 640 KByte Hauptspeicher – da bringt diese Einrichtung schon einen schönen Geschwindigkeitsvorteil.

Nun zu dem zweiten mit dem MS-DOS gelieferten Einheitentreiber, der »Erweiterten Ein-/Ausgabe-Unterstützung«. Der Treiber hat den Namen »ansi.sys«. Er wird nach demselben Mechanismus installiert wie die virtuelle Platte, so daß Ihre »config.sys« dann so aussehen könnte wie in Bild 1.

Der Aufgabenbereich des Ansi-Treibers ist vielfältig und eröffnet die verschiedensten Fähigkeiten

Die Aufgabe des Ansi-Treibers, wie man ihn auch nennt, ist es, Zeichen auf dem Weg zwischen Computer und Bildschirm beziehungsweise Tastatur abzufangen und umzudeuten. Das heißt zum Beispiel, daß eine Zeichenfolge, die von einem Programm oder Befehl (etwa vom Echo-Befehl) zum Bildschirm geschickt wird, dort gar nicht angezeigt wird, also unlesbar ist. Statt dessen schaltet diese Zeichenfolge aber etwa die Bildschirm-Farbe um oder verändert die Position des Cursors.

In gleicher Weise lassen sich alle Tasten der Tastatur umdefinieren, was natürlich nur bei Funktionstasten oder Tastenkombinationen sinnvoll ist. Dort kann es jedoch sehr nützlich sein, etwa einen oft benutzten, aber komplizierten Befehl auf eine Funktionstaste zu legen. Dieser Funktion bedienen sich allerdings nur Programme, die das MS-DOS für ihre Ein- und Ausgabe auch wirklich verwenden und nicht (wie etwa Lotus 1-2-3) das Betriebssystem umgehen und direkt auf das BIOS (»Basic Input Output System«) zugreifen. Hier also zwei Fragen: Erstens, wie macht man das? Zweitens, wozu ist das gut?

Das Funktionsprinzip ist einfach, in der Praxis aber nicht immer so leicht zu realisieren. Der Ansi-Treiber wird durch ein Esc-Zeichen und eine nachfolgende eckige Klammer »[« aktiviert. Findet er diese Zeichen, so interpretiert er die darauf folgenden Zeichen (Ziffern, Buchstaben und Semikolon zu ihrer Trennung) in ganz spezieller Weise. Wie das geschieht, ist im technischen Handbuch des MS-DOS beschrieben. Da aber die wenigsten dazu Zugang haben, finden Sie hier die wichtigsten Steuerzeichen. Tabelle 1 listet die Cursor-Steuerungszeichen auf. Der Unterschied zwischen Groß- und Kleinschreibung ist darin sorgfältig zu beachten. Das Kunststück ist nun die Erzeugung der mit »Esc« und »[« beginnenden Folge und ihr Transport zum Bildschirm. Programmierer haben es da einfach: In Basic zum Beispiel verwenden sie den Print-Befehl sowie »CHR\$(27)« für das »Esc« und »CHR\$(91)« für das »[« oder direkt die eckige Klammer (erzeugt durch [Alt-91]) in Anführungszeichen. Tippen Sie doch einfach einmal das kleine Programm von Listing 1 ein. Es erzeugt eine Stapelprozedur namens »farbe.bat«, mit der Sie die

Bild 1.
Inhalt der Konfigurations-Datei
(s. Bild 2)

```
C:> type config.sys
country=049
device=ansi.sys
device=vdisk.sys
files=16
buffers=6
```


Bildschirmfarbe beliebig umschalten können. Ihr Inhalt ist die Zeile

```
echo »Esc« [%1;%2m
```

Wenn Sie sie mit der Tabelle 2 verwenden, so wird durch die Eingabe von

```
color 32 40
```

der Echo-Befehl

```
echo »Esc« [32;40m
```

erzeugt und damit der Bildschirm auf »grün auf schwarz« umgeschaltet. Jede andere Hintergrundfarbe als schwarz sieht unschön aus, wie Sie leicht ausprobieren können. Kombinationen wie »31 41« vermeiden Sie aus einsehbareren Gründen. Sie könnten ebenso die Steuersequenzen in einer normalen Textdatei (nicht in einer Stapeldatei) unterbringen und sie mit »type« am Bildschirm anzeigen lassen. Die Stapel- wie auch die Textdatei lassen sich auch ohne Basic erzeugen, wenn Ihr Editor das Zeichen »Esc« als Bestandteil des erzeugten Textes handhaben kann (wie etwa der Texteditor im »Sidekick«).

Aber wozu lange nach dem »Esc« suchen? Es steht ja als spezielles »Meta-Zeichen« im Prompt-Befehl zur Verfügung. Dessen Verwendung besitzt den Vorteil, daß nach jedem Programmaufruf die entsprechende Zeichenkette an den Bildschirm geschickt wird. Was auch immer ein Anwendungsprogramm mit Ihrer durch »farbe.bat« geänderten Bildschirmfarbe macht, Sie brauchen sie nicht durch erneute Eingabe nach Programmende wieder auf den von Ihnen gewünschten Wert zu setzen. Denn der Prompt-Befehl läuft dann automatisch ab. Seine Standardform ist bekanntlich

```
prompt $n$g
```

um das Standardlaufwerk (durch »\$n«) und das Zeichen »>« (durch »\$g«) anzuzeigen. Das Meta-Zeichen für »Esc«

ist »\$e«, und das Zeichen »[« erzeugen Sie durch [Alt-91]. Setzen Sie also die Farbumschaltung davor:

```
prompt $e[32;40m$n$g
```

Das bringt immer die gewünschte Bildschirmfarbe zurück. Welchen Nutzen könnte es haben, mit dem Ansi-Treiber den Cursor auf dem Bildschirm positionieren zu können? Nun, manche Perfektionisten finden es unschön, daß Batch-Prozeduren, deren Ablaufprotokoll mit der Angabe

```
echo off
```

in der ersten Zeile unterdrückt werden sollte, genau diesen Befehl aber noch am Bildschirm anzeigen. Was ist zu tun?

Etwas für Perfektionisten: Die Cursor-Position auf dem Bildschirm per Ansi-Treiber

Wenn die Anzeige »echo off« geschrieben ist, muß der Cursor eine Zeile nach oben bewegt und die Zeile mit der Anzeige darin gelöscht werden. Das soll mit Hilfe eines Echo-Befehls in der zweiten Prozedurzeile geschehen. Dieser aber würde mindestens ein »carriage return«, also einen »Zeilenvorschub« als letztes darstellen. Also muß der Cursor noch einmal eine Zeile nach oben, um schließlich an der Stelle zu landen, wo das erste »echo off« stand. So sieht also die benötigte Prozedur in ihrem Kopf aus:

```
echo off echo »Esc« [1A»Esc« [K»Esc« [1A
```

Es bewirkt das Gewünschte: Zeile hoch, Zeile löschen, Zeile hoch, Zeile hinunter. Auch hier können Sie wieder mit einem kleinen Basic-Programm (Listing 2) die benötigte Stapelprozedur erzeugen. Der Effekt ist verblüffend, besonders, wenn Sie noch eine weitere Echo-Zeile anfügen, die irgendeinen Text schreibt. Sie sehen das erste »echo off« kurz aufblitzen, dann verschwinden, und dann Ihren Text an derselben Stelle auftauchen.

Wer läuft so schnell
wie unser Fuchs?



- **FoxBASE+** ist ein Datenbankentwicklungssystem bestehend aus Interpreter und Compiler.
- **FoxBASE+** ist in der Syntax kompatibel zu dBase III Plus
- **FoxBASE+** kann 100 KByte Quellcode in weniger als 60s kompilieren.
- **FoxBASE+** läuft unglaublich schnell – es hat in allen Vergleichstests exzellente Ergebnisse erzielt.
- Der **FoxBASE+** Compiler versteht den kompletten Quellcode des Interpreters, zeitraubende Änderungen sind nicht notwendig.
- **FoxBASE+** existiert in einer Single-User Version und in einer netzwerkfähigen Multi-User-Version.
- **FoxBASE+** ist ab sofort in deutscher Sprache erhältlich.

Am Rohrbusch 79, 4400 Münster,
Telefon 02534/7093

ComFood
Software GmbH

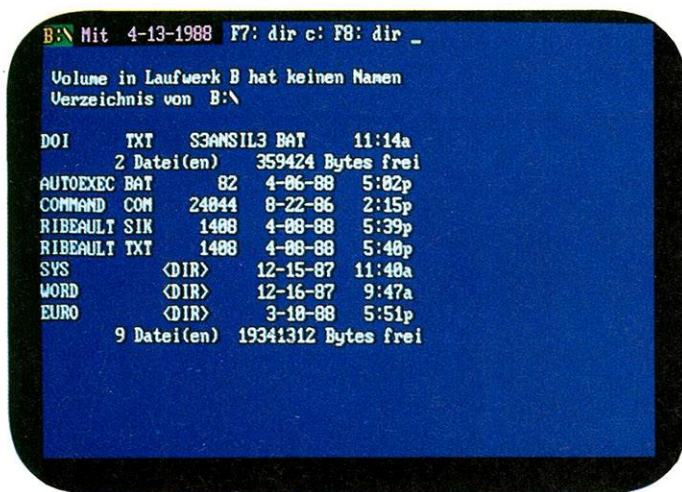


Bild 2. Einfache Menüs mit »ansi.sys«

Benutzen Sie den Ansi-Treiber auch, um Fehlermeldungen aus Stapelprozeduren deutlich hervortreten zu lassen. Eine Prozedurzeile wie

```
echo »Esc«[7m *** FEHLER *** »Esc«[0m
```

zeigt die Fehlermeldung in Umkehranzeige an.

Mit dem Ansi-Treiber läßt sich auch die Tastenbelegung ändern. Die entsprechende Esc-Zeichenfolge dafür ist

```
»Esc«[ #1; #2p
```

Darin steht »#1« für den ASCII-Code der Taste beziehungsweise den Tastatur-Suchcode. Beide Informationen sind im Anhang K des Basic-Handbuchs zu finden. Die wichtigsten Codes zeigt Tabelle 3; wichtig sind insbesondere die Funktionstasten und ihre Kombinationen mit [Shift], [Alt] und [Ctrl].

Die zweite Angabe »#2« ist die Nummer der Taste, die der ersten zugeordnet werden soll oder aber eine Zeichenkette in Anführungszeichen. Für die erste Angabe müssen Sie für Sondertasten (zum Beispiel Funktionstasten) zwei Zahlen angeben, von denen die erste immer »0« ist. Das hört sich sehr kompliziert an, und ein kleines Beispiel verdeutlicht die Sache: Die Funktionstaste [F7] soll mit dem Befehl »dir c:« zum Anzeigen des Festplatten-Inhaltes belegt werden (Bild 2). Die Funktionstasten beginnen mit dem Tastaturcode 59 für [F1] – [F7] hat also die Nummer 65. Die Angabe »#1« für die Sondertaste muß mit der Null beginnen, also als »0;65« verschlüsselt werden. Diese Angabe folgt auf das mit »\$« verschlüsselte Esc-Zeichen und das »[« (erzeugt durch [Alt-91]) im Prompt-Befehl.

Nach einem weiteren Semikolon als Trennzeichen kommt die gewünschte Zeichenkette in Anführungszeichen, also »"dir c:«". Dann erneut ein trennendes Semikolon und der Code »13« für die Enter-Taste. Ein »p« schließt die Sequenz ab und macht sie für den »ansi-Treiber« als Tastaturänderung kenntlich.

Der Prompt-Befehl zur Belegung der Funktionstaste »F7« mit dem Befehl »dir c:« sieht dann also so aus:

```
prompt $e[0;65;"dir c: ";13p
```

Soll der auf der Funktionstaste liegende Befehl nicht ausgeführt werden, sondern möchte man die Gelegenheit haben, ihn zu ergänzen, so läßt man die »13« weg. Um also einen ergänzbaren Copy-Befehl auf [F6] zu legen, geben Sie ein:

```
prompt $e[0;64;"copy "p
```

Danach können Sie durch

```
prompt
```

das Standard-Prompt »A>« wiederherstellen. Benutzen Sie diesen Mechanismus doch einmal dazu, zusätzlich zum Dir-Befehl in [F7] eine sortierte Dir-Anzeige auf [F8]

und auf [F9] die Rückkehr von einem beliebigen Unterverzeichnis zu legen. Das Ganze soll zusammen mit dem Datum in einer Statuszeile am Kopf des Bildschirms ständig zu sehen sein. Laufwerk und Unterverzeichnis sollen in rot auf grün, Datum in rosa, Funktionstasten-Belegung weiß auf blau erscheinen.

Eine bunte Stapelprozedur, die Klarheit schafft

Die entsprechende Stapelprozedur dazu sehen Sie in Listing 3. Die ersten drei Prompt-Befehle erledigen die Tastenbelegung. Der vierte Prompt-Befehl, ein wahres Ungewöhnliches in einer einzigen langen Zeile, ist wie ein Programm von links nach rechts zu lesen.

Nehmen Sie die Tabellen zur Interpretation zu Hilfe: Zuerst wird die gegenwärtige Cursor-Position gespeichert und dann der Cursor auf die Position »1;1« (Zeile 1, Spalte 1) gesetzt. Eventuell vorhandene Anzeigen, die am Bildschirm hochgerollt sind, werden ab dieser Position durch die Angabe »[K« gelöscht. Die Farbe wechselt nun zu »rot auf grün«, um mit dem Prompt-Zusatz »\$p« das gerade gültige Unterverzeichnis anzuzeigen. Nach der Farbumschaltung zu »rosa auf schwarz« folgt dann das Datum. Wenn Sie wollen, können Sie nach dem Datum (durch »\$d«) auch noch die Uhrzeit mit »\$t« protokollieren. Nun ein erneuter Farbwechsel zu »weiß auf blau« und die Anzeige der Funktionstasten-Belegung. Schließlich stellen Sie die Farbe auf »grün auf schwarz« zurück und kehren zur gespeicherten Cursor-Position zurück, wo die Eingabe-Aufforderung »>« angezeigt wird. Soll die Eingabe immer in der Fußzeile des Bildschirms erfolgen, so ersetzen Sie das »\$e[u« durch »\$e[25;1f«.

Zu beachten ist noch das »echo on« in der ersten Zeile der Prozedur. Es ist notwendig, weil die Ansi-Steuercodes zum Bildschirm gesendet werden müssen – zum Beispiel dadurch, daß die Prompt-Befehle als »Echo« angezeigt werden. Unterdrücken Sie dies, so wird nur der Prompt ständig geändert, aber nie wirksam – eine kleine, aber tückische Falle.

An diesen Beispielen haben Sie sicher die Einsatzmöglichkeiten der Einheitentreiber erkannt. Ihrer Experimentierfreude sind nun kaum noch Grenzen gesetzt. Sinnvolle und nützliche, vielleicht auch überraschende Anwendungen sind das Resultat.

(Jürgen Beetz/dm)

```
[NVA] 10 REM Farbe.bas: schreibt ANSI-Befehl
[LF9] 20 REM           mit ESC in in "Farbe.BAT"
[XA8] 30 OPEN "Farbe.BAT" FOR OUTPUT AS #1
[AS7] 40 PRINT #1,"ECHO "+CHR$(27)+"[1;2m"
[942] 50 SYSTEM
```

Listing 1. Basic-Programm zum Erzeugen einer variablen Stapelprozedur zum Umschalten der Bildschirmfarbe

```
[LGG] 10 REM Echo.bas: schreibt ANSI-Befehl mit ESC in
in "Kopf.BAT"
[FR7] 20 OPEN "Kopf.BAT" FOR OUTPUT AS #1
[9H4] 30 PRINT #1,"ECHO OFF"
[EPC] 40 PRINT #1,"ECHO "+CHR$(27)+"[1A"+CHR$(27)+"[K"+
CHR$(27)+"[1A"
[942] 50 SYSTEM
```

Listing 2. Basic-Programm zum Erzeugen eines Stapelprozedur-Kopfes ohne »echo off«-Anzeige

```
[VK2] 0: echo on
[9E6] 1: prompt $e[0;65;"dir c: ";13p
[NP8] 2: prompt $e[0;66;"dir | sort /+14";13p
[2P5] 3: prompt $e[0;67;"CD\ ";13p
[G4U] 4: prompt $e[s$e[1;1f$e[K$e[31;42m$e[35;40m
$d $e[37;44m F7: dir c: F8: dir | sort /+14
F9: cd\ $e[32;40m$e[u$g
```

Listing 3. Prompt-Befehl zur Belegung von Funktionstasten und zur Anzeige in der Statuszeile

Cursor-Aktion	Code und Erläuterung
positionieren	»Esc«[# 1; # 2f # 1 = Zeile, # 2 = Spalte
speichern	»Esc«[s, gegenwärtige Position (Zeile/Spalte) wird gespeichert
wiederherstellen	»Esc«[u, letzte gespeicherte Position wird wiederhergestellt
löschen Zeile	»Esc«[K löscht Zeile ab (incl.) Cursor-Position
aufwärts	»Esc«[# 1A, # 1 = Zahl der Zeilen
abwärts	»Esc«[# 1B, # 1 = Zahl der Zeilen
vorwärts	»Esc«[# 1C, # 1 = Zahl der Spalten
rückwärts	»Esc«[# 1D, # 1 = Zahl der Spalten

Tabelle 1. Cursor-Steuerungssequenzen des »ansi-Treibers

Code	Erläuterung
	»Esc«[# 1; # 2m # 1 = erste Angabe, # 2 = zweite Angabe
30	Vordergrund schwarz
31	Vordergrund rot
32	Vordergrund grün
33	Vordergrund gelb
34	Vordergrund blau
35	Vordergrund rosa
36	Vordergrund türkis
37	Vordergrund weiß
40	Hintergrund schwarz
41	Hintergrund rot
42	Hintergrund grün
43	Hintergrund gelb
44	Hintergrund blau
45	Hintergrund rosa
46	Hintergrund türkis
47	Hintergrund weiß
	»Esc«[# 1m (nur eine Angabe)
0	normal (weiß auf schwarz)
1	hell
4	unterstrichen (nur bei monochrom)
5	blinkend
7	Umkehranzeige (schwarz auf weiß)
8	unsichtbar (Vorsicht!)

Tabelle 2. Steuerungssequenz und Codes für Bildschirmfarben

Code-Nr.	Taste
1	Escape
13	Enter
14	Backspace
29	Ctrl
56	Alt
59	F1
60	F2 bis
68	F10
84	Shift-F1 bis
93	Shift-F10
94	Ctrl-F1 bis
103	Ctrl-F10
104	Alt-F1 bis
113	Alt-F10
71	Home
72	Cursor Up
73	PgUp
75	Cursor Right
76	Leertaste
77	Cursor Left
79	End
80	Cursor Down
81	PgDn

Tabelle 3. Auswahl aus den Tastatur-Suchcodes

Keine Chance für Daten-Geier

Produkt des Monats
CHIP 3/88



SaveDir

versteckt Ihre Programme und Daten

Versteckte Programme und Daten sind optimal geschützt. Niemand erfährt, was Sie gespeichert haben. Damit löst SaveDir ein Problem, das jeder PC-Benutzer hat.

SaveDir versteckt extrem schnell: In 8 Sekunden sind 10 MB auf einem 6 MHz-AT versteckt.

Sie allein entscheiden, was versteckt werden soll.

Nur mit SaveDir und Ihrem persönlichen Passwort ist es möglich, Verstecktes wieder sichtbar zu machen.

SaveDir läuft auf XT/AT-PC's ab DOS Version 2.0.

SaveDir wird mit deutschem Handbuch geliefert. Menü-Führung in deutsch. Kein Kopierschutz.

Auch der Preis ist sensationell: nur DM 198,-
Überzeugen Sie sich selbst.

Sichern Sie sich Ihre SaveDir-Demo. Jetzt.



Andreas Müller Software

Dieffenbachstraße 59 · 1000 Berlin 61
Telefon (030) 691 6614

SaveDir-Coupon

Name _____

Straße _____

PLZ, Ort _____

Telefon _____

Datum und Unterschrift _____

☐ Ich bestelle SaveDir für nur DM 198,-

☐ Ich bestelle die Demo für nur DM 19,80 (Anrechnung DM 10,- bei Kauf)

☐ Bitte Info zusenden

☐ Verrechnungsscheck beigefügt

☐ Nachnahme + DM 8,- für Versand

DOS Extra

MS-DOS-Umgebung und Batch-Variable

Der Batch-Sprache von MS-DOS fehlen Eingabemöglichkeiten, ein schon oft beklagter Mißstand. Eine einfache, in ihrer Leistung aber auch beschränkte Möglichkeit, dies zu beheben, liegt in der Benutzung des Errorlevels.

Sie haben sicher schon einmal versucht, in Ihrer Datei »autoexec.bat« Wahlmöglichkeiten einzubauen. Man schreibt beispielsweise mit Turbo Pascal ein Programm, das die Tastatur abfragt und in Abhängigkeit von der gedrückten Taste einen Endecode setzt (mit Hilfe der Turbo Pascal-Funktion »halt«). In Abhängigkeit von diesem Endecode können Sie dann mit der Anweisung »if...errorlevel« verzweigen.

Diese Methode liefert zufriedenstellende Ergebnisse, wenn es zum Beispiel darum geht, in einem Batch Auswahlmöglichkeiten anzubieten. Typisches Beispiel: Ein Start-Menü bei einem Computer mit einer Festplatte.

In manchen Situationen möchte man aber interaktiv, also während eines Batchlaufes, frei definierbare Variablen mit Werten belegen und diese Variablenwerte weiter verarbeiten. Dazu braucht man Speicherplatz, in denen man die Variablen speichert und eine Möglichkeit, Werte einzugeben und später abzufragen.

In der Ausgabe DOS International 2'88 wurde auf eine solche Fähigkeit hingewiesen. Das dort vorgestellte Assemblerprogramm erlaubt die interaktive Eingabe eines Wertes für eine Variable und ihre Weiterverarbeitung in einem Batch.

Den Umgebungsbereich legt MS-DOS beim Starten des Computers an

Dieses Programm, wie auch das hier vorgestellte namens »setumg«, benutzt dafür den MS-DOS-Umgebungsbereich. Für den in der Assemblerprogrammierung unerfahrenen Leser macht es einige Mühe zu verstehen, wie das Programm arbeitet. Deshalb hier ein entsprechendes Turbo Pascal-Programm mit dem Namen »setumg«, dessen Arbeitsweise detailliert erläutert ist.

MS-DOS legt beim Systemstart einen sogenannten Umgebungsbereich (Environment) an. Dieser Bereich enthält – im Klartext – spezielle MS-DOS-Variablen wie »comspec«, »path« und »prompt«. Der Inhalt einer solchen Umgebung kann beispielsweise so aussehen:

```
comspec=c:\command.com
path= \ ; \system; \util
prompt=$p$g
```

Diese MS-DOS-Variablen können Sie in einem Batch-Programm abfragen. Dazu schließt man den Variablennamen in %-Zeichen ein. In einem Batch bewirkt beispielsweise die Zeile

```
echo %comspec%
```

die Ausgabe des Pfades und des Namens des Kommandointerpreters, hier also »c:\command.com«.

MS-DOS sieht eine Eingabemöglichkeit für den Umgebungsbereich mit dem Befehl »set« vor. »set« ohne Parameter listet den Inhalt der Umgebung aus (wie oben). Aber mit einer Anweisung vom Typ

```
set Variablenname=Wert
```

wird eine MS-DOS-Variable der Umgebung aktualisiert oder, falls noch nicht vorhanden, neu eingetragen (vorausgesetzt, der definierte Platz reicht dazu!). »set X = ABC« würde in obigem Beispiel also die Zeile X = ABC in den Umgebungsbereich neu aufnehmen.

Diese Eingabemöglichkeit nützt natürlich innerhalb eines Batch-Programms wenig, weil Sie ja mit »set« nicht interaktiv Werte übergeben können. Das Problem löst das Programm »setumg«: Beim Aufruf wird der Variablenname als Parameter übergeben. Das Programm fragt nach dem Variablenwert und fügt direkt einen entsprechenden String in die MS-DOS-Umgebung ein. Was charakterisiert nun diesen Umgebungsbereich? Der Umgebungsbereich ist ein Klartextbereich. Die einzelnen Eintragungen sind durch das Zeichen mit dem ASCII-Code 0 voneinander getrennt. Das aktuelle Ende des Umgebungsbereichs kennzeichnen zwei aufeinanderfolgende ASCII-Nullen.

Es gibt Unterschiede in der Größe des Umgebungsbereichs bei den verschiedenen MS-DOS-Versionen

Der Umgebungsbereich wird beim Starten des Systems angelegt. Seine Größe begrenzt MS-DOS (ab der Version 3.1) automatisch auf 160 Byte. Dies läßt sich aber durch entsprechenden Aufruf des »command.com« ändern. Die obere Grenze beträgt bei MS-DOS 3.2. und 3.3 32 KByte, bei MS-DOS 3.1 jedoch nur 992 Byte, das sind 62 Paragraphen zu je 16 Byte. Um die Größe der Initialisierung zu ändern, muß man die Datei »config.sys« entsprechend modifizieren. Mit der Zeile

```
shell=C:\command.com /e:240 /p
```

wird die Datei »command.com« von »C:\« geladen. Der Parameter »/e:240« (bei MS-DOS 3.2 und 3.3) legt die Größe des Umgebungsbereichs auf 240 Bytes fest. Bei MS-DOS 3.1. steht hier anstelle der Bytezahl die Anzahl der sogenannten Paragraphen. Dieser ergibt sich aus der Bytezahl dividiert durch 16, bei 240 also den Wert 15. Ohne die Eintragung »/p« wird die Datei »autoexec.bat« nicht mehr automatisch geladen. »/p« installiert »command.com« jedoch so, daß auch der obere Teil des »command.com« speicherresident bleibt. Dies ist bei den heute üblichen Speicherkapazitäten jedoch kein Nachteil mehr. Wenn nach dem Anlegen des Umgebungsbereichs speicherresidente Programme installiert sind (das ist ja die Regel), so kann man den Umgebungsbereich nachträglich nicht mehr ausdehnen.

Der Befehl »errorlevel« bietet nur begrenzten Eingriff in einen Batch

Der effektive Speicherort des Umgebungsbereichs und die voreingestellte maximale Länge verwaltet MS-DOS an nicht dokumentierten Orten im Speicher. Wie kommt man nun an diese Informationen heran? MS-DOS trägt die Segmentadresse der Umgebung beim Laden des »command.com« in den Arbeitsspeicher ein. Sie brauchen daher zunächst Informationen über den Ladeort von »command.com«. Insbesondere müssen Sie innerhalb des Programms »setumg« diesen Speicherplatz aufspüren. Diese Informationen erhalten Sie im »Program Segment Prefix« (siehe Bilder 1 bis 6).

Jedes geladene Programm mit der Dateierweiterung »com« erhält diesen 256-Byte-Vorspann, abgekürzt PSP. Das PSP beginnt bei Offset 0 des Codesegments des Programms und beinhaltet zwei wichtige Informationen: In Offset 2Chex und 2Dhex (dezimal 44, 45) befindet sich ein Zeiger auf eine Kopie der MS-DOS-Umgebung, die für das Programm eigens angelegt wird. Diese Kopie existiert jedoch nur, so lange das Programm läuft. Eine dauerhafte Änderung der Umgebung resultiert aus der Änderung dieser Kopie nicht. Außerdem wird jedes Programm, außer natürlich »command.com«, von einem anderen MS-DOS-Programm, dem sogenannten Eltern-Prozeß, geladen. Normalerweise übernimmt »command.com« die Aufgabe, Anwenderprogramme zu starten. Damit man ein Programm

korrekt beenden kann, enthält das PSP des Programms die Segmentadresse des Eltern-Prozesses, meistens die Segmentadresse des »command.com«. Diese Segmentadresse, Prozeßidentifikation (PID) genannt, steht im Offset 16hex und 17hex (dezimal 22, 23) des PSP. Mit diesen Informationen können Sie nun die Adresse des Original-Umgebungsbereichs finden.

Bei MS-DOS 3.3 funktioniert dies auf eine naheliegende Weise: Auch die Datei »command.com« wird mit einem PSP geladen. Dort zeigt aber die Adresse an der Stelle 2Chex, 2Dhex auf die Original-Umgebung. Daraus ergibt sich für die Suche nach der Adresse der Umgebung in unserem Turbo-Pascal-Programm »setumg« folgendes:

Die Segmentadresse von »setumg.com« muß bestimmt werden. Dazu verwendet man die Turbo-Pascal-Funktion »cseg«.

Man muß PID, also die Codesegmentadresse von »command.com« definieren, die Sie in dem Wort »PID:= memw[Cseg:\$16]« finden.

Die Segmentadresse »segumg« der Umgebung, die dem »command.com« zugeordnet ist, muß ebenfalls noch bestimmt sein. Für MS-DOS 3.3 gilt hier »segumg:= memw [PID:\$2C]«. Das erste Byte der Umgebung steht dann an der Adresse »[segumg:0]«.

Leider ist die Situation bei früheren MS-DOS-Versionen komplizierter. Dort findet sich an der Stelle im PSP des »command.com« eine 0, an der eigentlich auf den Umgebungsbereich gezeigt werden sollte. Hier stehen die relevanten Informationen in dem (sonst undokumentierten) 16-Byte-Bereich, der dem PSP vorangeht. Der Wert des 3.Byte dieses Bereichs, nennen wir ihn »z«, liefert die nötige Information: »PID + 1 + z« ergibt die Adresse des Umgebungssegments. Das bedeutet, daß die Umgebung (z + 1) * 16 Bytes hinter dem Beginn des Codesegments von »command.com« beginnt.

Mit »setumg.com« können Sie in einer Batch-Datei interaktiv arbeiten

Die Pascal-Funktion (Listing 1) trägt beiden Fällen Rechnung und ermittelt die korrekte Segmentadresse der Umgebung in allen getesteten Versionen 3.1, 3.2 und 3.3 von MS-DOS beziehungsweise PC-DOS.

```
[PF8] 0: Function SegUmg:Integer;
[535] 1: var PID,s:integer;
[RH2] 2: z:byte;
[BH1] 3: begin
[LB5] 4:   PID:=memw[Cseg:$16];
[OD4] 5:   s:=memw[PID:$2C];
[E82] 6:   if s=0 then
[BH1] 7:     begin
[SA4] 8:       z:=mem[PID-1:3];
[E22] 9:       s:=PID+z+1;
[TU0] 10:    end;
[GN2] 11:   SegUmg:=s;
[TU0] 12: end;
```

Listing 1. Mit dieser Funktion erhalten Sie die Umgebungsadresse

Die maximale Länge des Umgebungsbereichs ist analog dem 16-Byte-Bereich vor der MS-DOS-Umgebung zu entnehmen. Die Funktion in Listing 2 ermittelt diese Länge.

```
[TN8] 0: function UmgLaenge:Integer;
[BH1] 1: begin
[EG8] 2:   UmgLaenge:=mem[SegUmg-1:3]*16;
[TU0] 3: end;
```

Listing 2. So erfahren Sie die Länge des Umgebungsbereichs

1. Beispiel: PC-DOS 3.3

(a) Anfang des PSP von SETUMG

Adresse	Hexcode	PID
SEG:OFFS	0 1 2 3 4 5 6 7 8 9 A B C D E F	
16D7:0000	CD 20 00 A0 00 9A F0 FE 1D F0 2F 01 6E 0F 3C 01	
16D7:0010	6E 0F 56 05 6E 0F 6E 0F 01 01 01 00 02 FF FF FF	

Das PSP enthält also an den Offsets 16,17 das Wort 6E 0F, womit nach Umstellung gemäß Intelkonvention sich **PID = 0F6E** ergibt.

Bild 1. Ein Teil des PSP des Programms »setumg«

(b) Anfang des PSP von COMMAND.COM

Adresse	Hexcode	SegUmg
SEG:OFFS	0 1 2 3 4 5 6 7 8 9 A B C D E F	
0F6E:0000	CD 20 00 A0 00 9A F0 FE 1D F0 B2 02 6E 0F 3C 01	
0F6E:0010	6E 0F 56 05 6E 0F 6E 0F 01 01 01 00 02 FF FF FF	
0F6E:0020	FF FF FF FF FF FF FF FF FF FF 5A 10 25 08	

Die Segmentadresse des Umgebungsbereichs ist also **SegUmg=105A**.

Bild 2. Der PSP von »command.com«

Nun besteht die Arbeit des Programmes »setumg« nur noch darin, den Umgebungsbereich in einen Turbo-Pascal-String einzulesen, diesen String zu modifizieren und ihn wieder rückzuspeichern. Das Programm »setumg« geht so vor:

Die Hauptarbeit leisten die Funktionen »segumg« sowie die Prozeduren »getumgebung« und »putumgebung«. »segumg« liefert die Segmentadresse des Umgebungsbereichs, »getumgebung« liest den Umgebungsbereich in den String »umgstr« ein. »putumgebung« speichert den modifizierten String zurück. Durch die Übernahme der Umgebung in einen Turbo-Pascal-String ergibt sich eine Beschränkung auf maximal 255 Byte. Mit einem eindimensionalen Byte-Array läßt sich das Programm so modifizieren, daß auch größere Umgebungen verarbeitet werden können.

Mit »setumg.pas« bekommen Sie den Umgebungsbereich in den Griff

Das Programm »setumg« (Listing 3) rufen Sie mit »setumg Variablenname Kommentar« auf. Beispielsweise mit »setumg X Bitte geben Sie einen Wert für X ein«. Der erste übergebene Parameter wird als Variablenname festgelegt, die restlichen Parameter als Kommentarart angezeigt, wenn Sie der Variablen einen Wert zuweisen. Ist kein Parameter angegeben, so endet das Programm mit dem Ende-code 1. Sonst untersucht das Programm, ob die Variable bereits in der Umgebung vorkommt.

(c) Anfang des Umgebungsbereichs

SEG:OF	Hexcode	ASCII-Code
0	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF
105A:00	43 4F 4D 53 50 45 43 3D 43 3A 5C 43 4F 4D 4D 41	COMSPEC=C:\COMMA
105A:10	4E 44 2E 43 4F 4D 00 50 41 54 48 3D 5C 3B 4C 53	ND.COM PATH=\\S
105A:20	59 53 54 45 4D 5C 4D 53 44 4F 53 3B 5C 53 59 53	SYSTEM\MSDOS\SYS

usw.

Bild 3. Der Beginn des Environments

(c) Anfang des Umgebungsbereichs

SEG:OF	Hexcode	ASCII-Code
0	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF
105A:00	43 4F 4D 53 50 45 43 3D 43 3A 5C 43 4F 4D 4D 41	COMSPEC=C:\COMMA
105A:10	4E 44 2E 43 4F 4D 00 50 41 54 48 3D 5C 3B 4C 53	ND.COM PATH=\\S
105A:20	59 53 54 45 4D 5C 4D 53 44 4F 53 3B 5C 53 59 53	SYSTEM\MSDOS\SYS

usw.

Bild 4. Der Anfang des PSP von »setumg« bei PC-DOS 3.2


```

[DH5] 0: Program setupmg;
[000] 1:
[LS3] 2: const Farbe=7;
[NV3] 3: UmgLaenge=160;
[000] 4:
[8D7] 5: type String255=String[255];
[000] 6:
[90F] 7: var UmgStr, VarName, Wert, StrAnf, StrEnde: string255;
[4HA] 8: i, p, VarPos, Laenge, Videomod: byte;
[000] 9:
[G4C] 10: function Gross(x:string255):string255;
[XG4] 11: var i: integer;
[9F3] 12: s:string255;
[BH1] 13: begin
[Z21] 14: s:='';
[10D] 15: for i:=1 to Length(x) do s:=s+UpCase(copy(x,i,1));
[5E2] 16: Gross:=s;
[TU0] 17: end;
[000] 18:
[PF8] 19: function SegUmg: Integer;
[535] 20: var PID, s: integer;
[RH2] 21: z: byte;
[BH1] 22: begin
[LB5] 23: PID:=memw[Cseg:$16];
[0D4] 24: s:=memw[PID:$2C];
[E82] 25: if s=0 then
[BH1] 26: begin
[SA4] 27: z:=mem[PID-1:3];
[E22] 28: s:=PID+z+1;
[TU0] 29: end;
[GN2] 30: SegUmg:=s;
[TU0] 31: end;
[000] 32:
[TYB] 33: procedure GetUmgebung(Var U:String255);
[3F6] 34: var laenge,i: integer;
[BH1] 35: begin
[121] 36: U:='';
[8FE] 37: for i:=0 to UmgLaenge-1 do U:=U+char(mem[SegUmg+i]);
[1X4] 38: laenge:=pos(#0+#0,U);
[BUS] 39: U[0]:=chr(lo(laenge));
[TU0] 40: end;
[000] 41:
[STA] 42: procedure PutUmgebung(U:String255);
[T74] 43: var i: integer;
[BH1] 44: begin
[296] 45: for i:=0 to length(U)-1 do mem[SegUmg+i]:=ord(copy(U,i+1,1));
[TU0] 46: end;
[000] 47:
[BH1] 48: begin
[VWD] 49: {Für Farbsystem Textfarbe auf farbe setzen}
[0V6] 50: videomod:=mem[$40:$49];
[LMA] 51: if videomod=3 then TextColor(farbe);
[4M5] 52: {Variable suchen}
[B28] 53: if ParamCount=0 then halt(1);
[XH8] 54: VarName:=Gross(ParamStr(1));
[WZ5] 55: GetUmgebung(UmgStr);
[567] 56: laenge:=length(UmgStr);
[H49] 57: VarPos:=pos(VarName+'=',UmgStr);
[U5D] 58: {Alten Wert von Var in Umgebung tilgen}
[5YK] 59: if VarPos>0 then {Variable bereits vorhanden, Tilgung nötig}
[BH1] 60: begin
[HUC] 61: StrEnde:=copy(UmgStr,VarPos,Laenge-VarPos);
[MP8] 62: StrAnf:=copy(UmgStr,1,VarPos-1);
[375] 63: p:=pos(#0,StrEnde);
[GV5] 64: if p=0 then StrEnde:=''
[WZD] 65: else StrEnde:=copy(StrEnde,p+1,length(StrEnde)-p)+#0;
[GA7] 66: UmgStr:=StrAnf+StrEnde;
[TU0] 67: end;
[YH8] 68: {Eingabe Variablenwert}
[CUD] 69: for i:=2 to ParamCount do write(ParamStr(i),' ');
[7H3] 70: readln(Wert);
[ZME] 71: {Zusammenbau Umgebung und Übertragung in RAM}
[XZA] 72: UmgStr:=UmgStr+VarName+'='+wert+#0+#0;
[BFC] 73: if length(UmgStr)>UmgLaenge then halt(2);
[516] 74: PutUmgebung(UmgStr);
[5T0] 75: end.

```

Listing 3. Mit dem Programm »setumg« können Sie das Environment manipulieren

(b) 16 Bytes vor PSP und Anfang des PSP von COMMAND.COM

Adresse	Hexcode
SEG:OFFS	0 1 2 3 4 5 6 7 8 9 A B C D E F
0A68:0000	4D 69 0A D7 00 00 00 00 00 00 00 00 00 00 00 00
0A69:0000	CD 20 00 A0 00 9A F0 FE 1D F0 88 02 69 0A 3C 01
0A69:0010	69 0A EB 04 69 0A 69 0A 01 01 01 00 02 FF FF FF
0A69:0020	FF FF FF FF FF FF FF FF FF FF FF FF FF 00 00 45 07

Man sieht, daß der Zeiger auf den Umgebungsbereich hier mit 0000 besetzt ist. In Byte 3 des 16-Byte Bereichs vor dem PSP (Adresse 0A68:0003) steht D7. Die Segmentadresse des Umgebungsbereichs ist also 0A69+1+D7 = 0B41.

Bild 5. Die Werte des PSP am Anfang von »command.com«

(c) Anfang des Umgebungsbereichs

SEG:OF	Hexcode	ASCII-Code
0	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF
105A:00	43 4F 4D 53 50 45 43 3D 43 3A 5C 43 4F 4D 4D 41	COMSPEC=C:\COMMA
105A:10	4E 4A 2E 43 4F 4D 00 50 41 54 48 3D 5C 3B 4C 53	ND.COM PATH=\; \S
105A:20	59 53 54 45 4D 5C 4D 53 44 4F 53 3B 5C 53 59 53	SYSTEM\MSDOS;\SYS

Bild 6. Die Werte des Environments

Falls ja, wird »umgstr« in zwei Teile zerlegt: »stranf« und »strende«, also denjenigen Teil, der der Variablen in der Umgebung vorausgeht und denjenigen, der ihr folgt. Diese beiden Teile setzt das Programm zusammen, so daß letztlich der zu der Variablen gehörende Teil gelöscht ist. Dann weisen Sie über die Anweisung »read« der Variablen einen Wert zu und die zugehörige Zeichenfolge »Variablenname = Wert« wird an den »umgstr« angehängt.

Falls der nun entstehende String zu lang ist, endet das Programm mit dem Endcode 2. Der Umgebungsbereich bleibt dann unverändert. Die Endcodes können Sie mit »if ... errorlevel« abfragen. Damit die Meldungen des Programms nicht in dem für Turbo Pascal typischen Gelb auf dem Bildschirm erscheinen (nur bei Farbsystem), setzten Sie die Anzeigefarbe auf 7, also weiß. Das läßt sich durch Ändern der Konstantenvereinbarung »Farbe = 7« leicht den eigenen Wünschen anpassen. Die Konstantenvereinbarung für »umglaenge« können Sie auch durch eine andere oder durch die oben beschriebene Funktion »umglaenge« ersetzen.

```

[3H2] 0: echo off
[MM0] 1: cls
[SB5] 2: echo Test von SETUMG
[LE1] 3: echo .
[LS7] 4: echo aktuelle DOS-Umgebung:
[TM0] 5: set
[FF9] 6: echo Eintragen einer Variablen X:
[ZW2] 7: setumg x x=
[4EA] 8: echo Sie haben für x eingegeben: %x%
[XY9] 9: echo Das ist die neue DOS-Umgebung:
[TM0] 10: set

```

Listing 4. Diese Batch-Datei testet die Umgebungsvariablen

Das kleine Batchprogramm (Listing 4) liefert Ihnen ein Testprogramm, um die Änderungen zu prüfen.

DOS-EXTRA DATABOX

– Die Software zur DOS-EXTRA Nr. 3 –

**auf 2 Disketten im Format 5 1/4" oder
1 Diskette im Format 3 1/2"**

sagenhaft günstig, extra viel gute Software für's 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. 3

- Datenfernübertragungsprogramm in 100% Assemblercode
- Hercules-Grafik-Toolbox in Turbo Pascal
- Grafikfunktionen für Turbo C
- Zeichensatzeditor für Epson-kompatible Drucker
- Treiber für AT-Tastaturen
- alle begleitenden Listings zum Kurs »PC-glasklar«
- Demo für Sprachdigitalisierung

... und alle weiteren im Heft enthaltenen Listings

2 Disketten 5 1/4"
1 Diskette 3 1/2"

(Bestell-Nr. 357)
(Bestell-Nr. 358)

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, –

DMV-Verlag · Postfach 250 · 3440 Eschwege

– Bitte Bestellkarte benutzen –

Interrupts und ihre Handhabung beim PC

Vielleicht stellte sich auch Ihnen schon einmal die Frage, wie die Tastatur Ihres PC eigentlich funktioniert oder warum DOS über die genaue Uhrzeit Bescheid weiß. Alle diese Dinge erledigen auf Ihrem PC die verschiedenen Interrupts, eine besondere Art der Programmbearbeitung durch den Mikroprozessor.

Ursprünglich wurden Interrupts als eine schnelle Antwort des Prozessors auf ein – meistens von einem Peripheriegerät kommendes – Signal verwendet. So sendet zum Beispiel der Drucker ein Signal »ich bin bereit für neue Daten« an den Computer. Der Mikroprozessor unterbricht daraufhin den normalen Programmablauf und verzweigt in eine spezielle Routine. In dieser Routine werden zunächst die Registerinhalte gesichert und daraufhin neue Daten zum Drucker gesendet. Zum Abschluß veranlaßt ein spezieller Befehl die Rückkehr zum normalen Programmablauf. Ein Interrupt ist also im Prinzip nichts anderes, als ein Hardware-gesteuerter Unterprogramm-Aufruf. Dieser Mechanismus läßt sich auch abschalten. Das heißt, der Prozessor empfängt dann zwar einen Interrupt, unterbricht aber das laufende Programm nicht. Dieses Sperren oder Zulassen steuert ein bestimmtes Bit im Prozessorstatusregister. Ein ankommendes Interruptsignal wird über ein logisches »und« mit diesem Bit verknüpft.

Im großen und ganzen gilt der gerade geschilderte Mechanismus auch für den PC, hier sind jedoch 256 mögliche Interrupts vorgesehen. Die ersten 1024 Byte im Hauptspeicher eines PC sind für diese 256 Interruptvektoren reserviert. Da beim Intel-Prozessor 8088/8086 eine Speicheradresse in der Form »Segment: Offset« angesprochen wird, sind pro Interrupt vier Byte vorgesehen: zwei für den Offset und zwei für das Segment der Interruptroutine, auf die der jeweilige Vektor zeigt.

Ein PC kennt immer zwei Arten von Interrupts. Einmal die gerade beschriebenen, durch Hardware ausgelösten Interrupts, und dann die sogenannten Software-Interrupts. Letztere sind Interrupts, die das Programm selbst auslöst. Dabei spielt der Zustand des Interrupt-Flags, das die Interrupts sperrt oder zuläßt, keine Rolle. Stößt der

Prozessor bei der Programmbearbeitung auf einen solchen Befehl, wird aus dem ersten KByte des Hauptspeichers die jeweilige Adresse, genannt Interruptvektor, durch Multiplikation der Interruptnummer mit 4 berechnet und zu

dieser Interruptroutine verzweigt. Diese Software-Interrupts kann man noch unterteilen: Es gibt BIOS-Interrupts und DOS-Interrupts. Sie gewährleisten einen schnellen Zugriff auf sämtliche DOS-, beziehungsweise BIOS-

Nummer *	Adresse im Segment 0000: (hexadezimal)	Bemerkung:
0	20	Timer-Interrupt von Kanal 0 des Intervall-Timers (Intel 8253). Der Timer läuft mit 1,19318 MHz; dieser Interrupt wird also ungefähr 18,2 Mal in der Sekunde bei einem Teillingsfaktor von 65536 aufgerufen. Mit diesem Interrupt realisiert das BIOS einen Betriebszeitgeber, der Aufschluß über die bisher vergangene Zeit seit dem letzten Start des Systems gibt. Dieser Betriebszeitgeber wird vom MS-DOS benutzt, um so die Kontrolle über die aktuelle Uhrzeit zu haben. Am Schluß der Routine steht der Befehl »int 1Ch«; dieser Vektor zeigt normalerweise auf einen Iret-Befehl. Hier können eigene Interruptroutinen eingeklinkt werden. Zu beachten ist dabei, daß diese Interruptroutinen – wie alle anderen Interruptroutinen auch – sämtliche Register, die sie benutzen, sichern und mit einem Iret-Befehl beenden müssen.
1	24	Tastaturunterbrechung. Dieser Interrupt wird jedesmal aufgerufen, wenn eine Taste gedrückt oder losgelassen wurde. So erkennt der Rechner, ob zum Beispiel Shift-Tasten betätigt wurden, ein Sonderzeichen mittels der Alt-Taste und einer Zahl eingegeben wurde, etc. Empfängt die Tastatur ein Zeichen, so wird es in die BIOS-eigene Tastaturwarteschlange eingereiht. Ist diese voll, so ertönt der bekannte Piepser. Ebenso wird hier ein Warmstart mittels [Ctrl-Alt-Del] erkannt und ausgeführt. Diesen Interruptvektor verändert der deutsche Tastaturlreiber »keybgr«.
2	28	Die folgenden vier Interruptvektoren sind im PC-BIOS nicht belegt. In den meisten Nachbauten zeigen sie auf eine Routine, die feststellt, welcher von den vier möglichen Interrupts ausgelöst wurde und diesen dann sperrt. Der Interrupt mit der Nummer 2 kann jedoch bei vielen Kompatiblen, die mit einer batteriegepufferten Echtzeituhr ausgerüstet sind, durch Umstecken von Jumpers so konfiguriert werden, daß er – nach entsprechender Programmierung – zu einem, mit der Uhr vorher eingestellten Zeitpunkt, einen Interrupt auslöst. Er erfüllt also eine »Weckfunktion«. Die Interruptroutine muß dabei aber von der ursprünglichen BIOS-Routine auf eine eigene umgesetzt werden. Da im PC unterschiedliche Uhrenbausteine zum Einsatz kommen, empfiehlt sich ein Blick ins Handbuch (viele Hersteller verwenden jedoch den 58167 von NEC).
3	2C	Die folgenden beiden Interrupts sind den seriellen Schnittstellen COM1 und COM2 zugeordnet. Zu beachten ist hierbei, daß COM1 den Interruptvektor 4 und COM2 den Interruptvektor 3 belegt, also genau umgekehrt zur logischen Schnittstellenummerierung. Ein Interrupt wird von den Intel-Bausteinen 8250 (das sind die gebräuchlichen Bausteine für serielle Schnittstellen) ausgelöst, wenn ein Byte vollständig empfangen oder gesendet wurde oder ein Fehler bei der Übertragung auftrat. Auf diese Weise lassen sich serielle Empfangs- und Sendepuffer für die serielle Schnittstelle einfach und schnell implementieren. Wichtig ist, daß ein Interrupt an drei verschiedenen Punkten zugelassen werden muß: im 8250, im Interruptcontroller und unter den Adressen 2C oder 30, wo eine Interruptserviceroutine eingetragen sein muß, die die gewünschte Funktion realisiert.
4	30	Hier steht der Interruptvektor für COM1. Auch hier gilt das oben gesagte im Bezug auf Interruptprogrammierung. Soll eine Interruptverwaltung für die serielle(n) Schnittstelle(n) implementiert werden, so empfiehlt es sich, vorher einen Blick ins Handbuch zu werfen, und die Jumperstellungen mit den tatsächlich vorhandenen auf der Platine zu vergleichen. Dies erspart unter Umständen viele Stunden nervenaufreibender und fruchtloser Fehlersuche, da der vermeintliche Fehler eventuell gar nicht in der Routine, sondern in der Hardware selbst liegt.
5	34	Dieser Interruptvektor wird beim Starten des Systems von einem eventuell vorhandenen Festplatten-Controller belegt. Aus diesem Grund ist er auch beim Original-PC-BIOS nicht besetzt, da IBM Festplatten erst ab dem XT für ihre PCs vorsah. Die Funktion dieses Interrupts hängt stark vom verwendeten Controller ab; auch an dieser Stelle sollte das Controllerhandbuch zu Rate gezogen werden oder aber Sie informieren sich mittels einem Debugger über die Art des Controller-ROMs.
6	38	Disketten-Interrupt. Dieser Interrupt wird von dem Disketten-Controller (NEC 765) ausgelöst, wenn er ein Seek-, Recalibrate- oder Reset-Kommando erfolgreich ausgeführt hat. Da das BIOS nach einem solchen Kommando auf das Eintreffen dieses Interrupts wartet, führt ein Ausbleiben zu einem »Time Out«-Fehler, der in der MS-DOS-Fehlermeldung »Not ready error« resultiert.
7	3C	Dieser Interruptvektor ist dem Drucker zugeordnet. Beim Initialisieren des Systems setzt MS-DOS diesen Vektor auf eine einzelne Iret-Anweisung, also einen Dummy. Nach entsprechender Programmierung wird bei jedem »ACK«-knowledge-Signal des angeschlossenen Druckers ein Interrupt ausgelöst. So kann der Computer erkennen, wann der Drucker zur Aufnahme neuer Daten bereit ist. Dieser Interrupt findet jedoch kaum Verwendung.

* Zur Ermittlung der tatsächlichen Interruptnummer addiert man noch die Basisadresse (in diesem Falle 8) hinzu. Diese Nummer mit 4 multipliziert ergibt dann die Adresse des Interruptvektoreintrags im Segment 0000.

Tabelle 1. Hardware-Interrupts im PC

Interrupt-Nummer	Bemerkung:	
0	Timer Interrupt	*
1	Tastaturbestätigung	*
2	Interrupt vom 2. 8259A	
3	COM2 Interrupt	*
4	COM1 Interrupt	*
5	Interrupt für zweiten Drucker	
6	Diskcontrollerinterrupt	*
7	Interrupt für ersten Drucker	*
8	RTC Interrupt	*
9	Durchgang für Interrupt 2 vom ersten 8259-Baustein	
10	Reserviert	
11	Reserviert	
12	Reserviert	
13	80287 Coprozessor Interrupt	
14	Harddisk-Controllerinterrupt	*
15	Reserviert	

* Die Interrupts von 8 bis 15 werden vom zweiten 8259-Baustein im AT verwaltet. Ein »*« in der jeweiligen Zeile kennzeichnet einen Interrupt, der in dieser Form auch in einem PC oder XT vorhanden ist und dort ähnliche Funktionen erfüllt.

Tabelle 2. Interruptbelegung beim AT

Funktionen. Ferner lösen auch bestimmte Prozessorzustände einen Interrupt aus, zum Beispiel erzeugt eine Division durch 0 den Interrupt Nummer 0.

Dieser Beitrag befaßt sich jedoch lediglich mit den von der Hardware erzeugten Unterbrechungen. Dazu eine kurze Erklärung, wie ein von der Hardware ausgelöster Interrupt im PC behandelt wird. Es sind acht mögliche Interruptquellen vorhanden (Tabelle 1). Wird eine Unterbrechung ausgelöst, so gelangt diese zuerst zum Interruptcontroller (im PC der Baustein 8259 von Intel). Dessen Aufgabe ist es, die Interrupts zu verwalten, das heißt, Prioritätenvergabe, Interruptquittierung etc. zu erledigen. Ferner können schon im Interruptcontroller einzelne Interrupts gesperrt beziehungsweise zugelassen werden. Hat also ein Interrupt den Controller erfolgreich passiert, wird er an den Mikroprozessor weitergeleitet. Gleichzeitig gibt der Controller die Nummer des Interrupts an den Prozessor weiter. Genaugenommen addiert der Interruptcontroller vorher noch eine Basis hinzu, so daß die Interruptsteuerung flexibel bleibt.

Zur Verdeutlichung nehmen wir einmal an, daß der erste serielle Schnittstellen-Baustein (COM1) ein vollständiges Byte von der Schnittstelle gelesen hat. Daraufhin (vorausgesetzt, der Chip wurde vorher entsprechend programmiert) löst der Baustein einen Interrupt aus. Dieser Interrupt hat die Nummer 4. Im PC beginnen die Hardware-Interrupts bei der Nummer 8

(also ist 8 die eben erwähnte Basis). Der Interruptcontroller schickt nun die Interruptnummer 12 an den Prozessor. Dieser liest die Adresse 0:30hex (0:48dez) und verzweigt anschließend zur entsprechenden Interruptservice-Routine. In dieser Routine muß der Prozessor dem Interruptcontroller mitteilen, daß er den Interrupt erhalten und verarbeitet hat. Dies geschieht mit einem EOI (End Of Interrupt)-Signal, das an den Controller gesendet wird.

Dieser gerade beschriebene Mechanismus gilt jedoch nur für einen PC. Beim AT sieht die Sache anders aus: Hier sind zwei Interruptcontroller hintereinandergeschaltet. Theoretisch könnte man durch Kaskadierung 8x8 = 64 Interrupts kontrollieren, praktisch werden beim IBM-AT nur 16 Interrupts erkannt und verarbeitet, von denen einige Interruptvektoren nicht belegt sind. Tabelle 2 gibt die wichtigsten Unterschiede zwischen der AT- und XT-Interruptcontroller-Belegung wieder.

Für diejenigen Leser, die sich für die Funktionweise des Interrupt-Controllers im Detail interessieren, ist hier die Programmierung des 8259A im PC kurz beschrieben. Wichtig sind die beiden Port-Adressen 20 und 21hex. Liest man mit einer Eingabe-Anweisung (»in«) den Port 21hex, so erhält man die gerade zugelassenen Interrupts in einem Byte. Die aktiven Interrupts sind durch zurückgesetzte Bits gekennzeichnet. So bedeutet beispielsweise die Maske BChex folgendes:

- 1 – Druckerinterrupt: gesperrt
- 0 – Diskettencontroller-Interrupt: zugelassen
- 1 – Festplattencontroller-Interrupt: gesperrt
- 1 – COM1-Interrupt: gesperrt
- 1 – COM2-Interrupt: gesperrt
- 1 – RTC (Real Time Clock) Interrupt: gesperrt
- 0 – Tastaturbestätigung: zugelassen
- 0 – Timerinterrupt: zugelassen

Diese Maske läßt sich nun nach Belieben verändern. Es ist jedoch hier besondere Vorsicht angebracht, um nicht auf einmal »ohne Tastatur« vor dem PC zu sitzen, weil man aus Versehen die Tastaturbestätigung gesperrt hat. Ferner sollte die Zulassung des Interrupts erst nach dem Setzen des Interruptvektors und nach dem Aktivieren des Bausteines, der den Interrupt auslösen soll, erfolgen. So vermeiden Sie, daß zum Beispiel ein Baustein, der noch nicht initialisiert wurde, einen Interrupt erzeugen kann, der dann auch noch irgendwo im »Nichts« landet – weil der Interruptvektor nicht initialisiert wurde.

In der Interruptroutine selbst wird dann noch den Interrupt quittiert, damit der Interruptcontroller weiß, daß die Verarbeitung des Interrupt ordnungsgemäß abgelaufen ist. Am einfachsten geschieht dies mit dem Byte 20hex, das mit

```
MOV AL,20h
OUT 20h,AL
```

auf den Port 20hex geschrieben wird. Für den Interruptcontroller heißt das, daß der »zuletzt« angekommene Interrupt quittiert wird. Will man einen bestimmten Interrupt quittieren, so lautet das Byte 6xhex (01100xxx binär), wobei x für den Interrupt steht (0 für den Timer bis 7 für den Drucker), der quittiert werden soll. In der Praxis kommt jedoch meist die erste Lösung zum Zuge, da sie einfacher und flexibler ist.

Nun noch zu der Frage der jeweiligen Priorität eines Interrupts. So wie der 8259 im PC programmiert ist, kommt dem Interrupt Nummer 0 (also der Timer) die höchste Priorität zu, dem Drucker dagegen die niedrigste. Die Beschreibung der alternativen Prioritätenvergabe würde jedoch den Rahmen dieses Beitrags sprengen.

(Christoph Zimmermann/rg)

DOS-T3

AT-Tastatortreiber nach Maß

Eines der wichtigsten Hilfsprogramme für Personalcomputer ist der Tastatortreiber. Jedoch bedarf es einiger Programmierkenntnisse, um ihn an die eigenen Bedürfnisse anzupassen. Für AT-Benutzer gibt es jetzt den DOS-T3, einen Tastatortreiber in Assembler-Quellcode.

Für die Tastaturfunktionen stehen zwei Interrupts zur Verfügung: der INT09H (IRQ1) sowie der INT16H. Der Interrupt 09H wird jeweils nach Drücken oder Freigeben einer Taste erzeugt. Aus dem durch die Tastatur gelieferten Scancode zusammen mit bestimmten Kennzeichenbits wird dann das ermittelte Zeichen in den Tastaturpuffer geschrieben. Ein Eintrag in diesem Puffer besteht aus zwei Bytes: Bei einem normalen Zeichen aus dem Zeichenbyte und dem Scancode, bei erweitertem Code aus dem Nullbyte und dem Zeichenbyte. Die im Tastaturpuffer anliegenden Zeichen lassen sich mit Hilfe des Interrupts 16H testen und auslesen.

DOS-T3 übernimmt hier die Funktion der Interrupt-09-Routine. Für diese Aufgabe stehen DOS-T3 zehn Tabellen zur Verfügung, aus denen je nach Tastenkombination der richtige Wert ermittelt wird:

- Tastatur normal
- mit Umschalttaste
- mit ALT-Taste
- mit Ctrl-Taste
- mit Ctrl- und Umschalttaste
- mit Ctrl- und Alt-Taste
- Extended-Code für spezielle Tasten (Nummernblock und F1-F12)
- Akzenttabelle

Weiter bietet T3 je nach Geräteausstattung eine Cursorunterstützung sowie für Herculeskarten zusätzlich die Funktion »Bildschirm schwarz setzen«.

Cursorunterstützung

Über die Cursorunterstützung werden sich alle diejenigen freuen, die bereits die Erfahrung eines fehlenden Cursors nach Programmabsturz beziehungsweise Veränderung des Cursors durch ein Anwenderprogramm gemacht haben. Durch Drücken der Tastenkombination [Ctrl-Alt-Minustaste (grau)] wird der Cursor wieder eingeschaltet, beziehungsweise liefert ein nochmaliges Drücken einen vollen Cursorblock. Mit der Tastenkombination [Shift-Ctrl-Alt-Minustaste (grau)] wird der Cursor ausgeschaltet.

Bildschirm schwarz setzen

Gerade im Geschäftsbereich ist es häufig der Fall, daß ein bestimmtes Bild stundenlang auf dem Monitor flimmert. Dies führt auf Dauer dazu, daß sich das Bild in die Leuchtschicht des Monitors einbrennt. Hier hilft die im Tastatortreiber enthaltene Funktion »Bildschirm schwärzen«. Dabei wird nach Ablauf der vorgegeben Zeitdauer nach dem letzten Tastendruck das Register 6 des Bildschirmcontrollers mit dem Wert 0 beschrieben. Register 6 enthält die auf dem Bildschirm tatsächlich angezeigten Zeilen (vergleiche DOS Extra, Ausgabe 1/87, Seite 119ff). Zum Wiederherstellen des Schirminhaltes wird der aktive Modus über das Lichtgriffeltor ermittelt. Diese Funktion ist zunächst nur für die Herculeskarte verfügbar. Für die CGA-Karte

befindet sich bereits eine Version im Test. EGA wird erst mit einer späteren Ausgabe dieses Programmes unterstützt.

Der Treiber

DOS-T3 besteht aus vier Teilen: dem Datenbereich, der Routine zum Löschen des Bildschirms, dem eigentlichen Tastatortreiber sowie dem Installationsteil. Der Treiber wird mit einem Parameter zwischen 0 und 9 aufgerufen. Dieser Parameter dient zur Abfrage, ob und nach welcher Zeit der Bildschirm schwarz gesetzt werden soll.

Der Installationsteil

Hier wird der Tastaturinterrupt (INT 09H) und, bei einem Parameter zwischen 1 und 9, der Zeitgeberinterrupt (INT 1CH) verbogen. Anschließend wird das residente Programm verlassen oder, bei Parameterfehler beziehungsweise Zweitaufwurf, mit Fehlermeldung terminiert.

Der Datenbereich

Im Datenbereich finden sich unter anderem auch die Tabellen für den Tastaturcode. Diese lassen sich gemäß den eigenen Bedürfnissen ändern. Bis auf die letzten beiden sind alle Tabellen gleich aufgebaut und bestehen aus 88 Byte in der Reihenfolge des Scancodes. Die Belegung des Scancodes können Sie in der Regel dem MS-DOS-Handbuch entnehmen. Bei der Tabelle »Tabl_x« für den erweiterten Code ist bei Veränderungen Vorsicht geboten. Die letzte Tabelle besteht aus Gruppen zu je drei Byte. Das erste Byte ist für das Zeichen »'«, das zweite für »'« und das dritte für »↑« zuständig.

Zeitgeberinterrupt

Soweit beim Aufruf installiert (Parameter 1 bis 9), wird in der Routine »NEW1CH«, die 18,2 mal pro Sekunde aufgerufen wird, zunächst der Zähler »Screen_Count« um 1 erniedrigt. Ist Null erreicht, wird getestet, ob der Bildschirm bereits abgeschaltet ist und dies im negativen Fall nachgeholt. Danach erfolgt in beiden Fällen die Rückkehr zur ursprünglichen Vektoradresse des Interrupts 1CH.

Der Kern des Tastatortreibers

Der DOS-T3 Tastatortreiber besteht aus den folgenden Grundfunktionen:

- Zeichen einlesen und verarbeiten, gegebenenfalls Schirm einschalten
- der Routine »RT_ALT« für die Alt-(Zahl)-Eingabe
- der Umladeroutine »BOOT«
- der Druckeroutinen »CHK_Prt«, »Prt_Con1« für Parallelschaltung ([Ctrl-PrtSc]), »Hardcopy« zum Ausdruck des Bildschirms ([Ctrl-Shift-PrtSc])
- der Funktion »Pause« nach Drücken von [Ctrl-Num-Lock]
- der Funktion »Break« nach Drücken von [Ctrl-Scroll-Lock]
- der Funktion »Cursize« zum Ein-/Ausschalten des Cursors und zum Wechseln der Cursor-Größe

Beim Drücken und beim Freigeben einer Taste erfolgt jeweils der Aufruf des Interrupt 9H. Dieser zeigt nach Installation von T3 auf die Routine »NEW9H«. Hier wird zunächst die Tastatur gesperrt, damit kein weiterer Interrupt erfolgen kann, solange das anliegende Zeichen nicht verarbeitet ist. Dann wird anhand des Kennzeichen-Bytes »Screen_Off« getestet, ob der Bildschirm ausgeschaltet ist. Gegebenenfalls wird er wieder eingeschaltet.

Dabei kann es eventuell in besonderen Grafikmodi Schwierigkeiten geben. Jedoch liefen alle dem Autor bekannten Programme einwandfrei, insbesondere Word, das für solche Zwecke ein ausgezeichnetes Testwerkzeug ist, wie schon viele Programmierer zu ihrem Leidwesen erfahren mußten. Bei jedem Tastendruck wird überdies der

Zähler »Screen_Count« auf das installierte Maximum gesetzt. Danach wird der Tastencode über das Datentor 60H gelesen, ermittelt, ob eine Taste gedrückt oder losgelassen wurde und als nächstes auf spezielle Funktionstasten (FUNC_TEST) geprüft.

Da das Listing 1 – der Quellcode – ausreichend dokumentiert ist, ist es nur noch erforderlich, auf einige Besonderheiten einzugehen. Ein Bildschirmausdruck läßt sich nur durch Drücken der Tastenkombination [Shift-Ctrl-PrtSc] erreichen! Bei der Kombination [Ctrl-Alt-Del] wird zum Originalinterrupt im ROM-BIOS verzweigt. Die Kombination [(Shift)-Ctrl-Alt-Minustaste (grau)] ist bei einigen Kompatiblen mit der Taktfrequenzwahl belegt. Hier ist dann auch eine Verzweigung ins ROM-BIOS erforderlich. Das Programm ist insoweit anzupassen.

Je nach gedrückten Tasten, zum Beispiel Umschalt-, Ctrl-, Alt-Kombinationen, wird die zugrundeliegende Tabelle benutzt. Eine Ausnahme bildet die reine Alt-Tabelle. Hier findet ab »ALT_CNT« eine gesonderte Behandlung statt, da einige Programme die Originalbelegung als erweiterten Tastaturcode benötigen. Bei Word gibt es mit Alt-Kombinationen zusätzliche Schwierigkeiten.

Eine der längsten Routinen ist »AKZ_CHK« mit »CHAR_TST«. Hier ist jedes gültige Zeichen einzeln abzufragen und je nach vorausgegangenem Akzentzeichen das zusammengesetzte Zeichen aus der Tabelle »AKZ_TABL« zu entnehmen. Eine Besonderheit stellt die Subroutine »YZ_Test« dar, wo der Scancode der Tasten »y« und »z« vertauscht wird. Es fragen nämlich einige amerikanische Softwareprodukte den Scancode ab und nicht das Zeichen, so daß bei gefragtem »y« ein »z« einge-

geben werden mußte. Dies ist jetzt behoben. Ist dieser Austausch nicht erwünscht, so entfernen Sie den Aufruf in der Routine »COD_NORM«.

Die Routine »PUSH_IT« ist für das eigentliche Schreiben des Wortes in den Tastaturpuffer zuständig. Hier erfolgt auch die Abfrage, ob der Tastaturpuffer voll ist. Nach Eingabe eines gültigen Zeichens oder nach Anspringen einer der oben angegebenen Funktionen wird die Tastatur wieder zur nächsten Eingabe freigegeben.

Assemblieren des Treibers

Geben Sie das Programm als unformatierte Textdatei »T3.ASM« ein und assemblieren Sie es dann mit Hilfe des MASM Version 4.00 (eventuell Version 3.0) unter der Bezeichnung »MASM T3«. Anschließend rufen Sie den Linker mit »LINK T3« auf. Die Stackfehlermeldung des Linkers ignorieren Sie, da sie für die noch zu erzeugende COM-Datei unerheblich ist. Anschließend wird mit »exe2bin« eine Com-Datei erzeugt:

```
exe2bin T3.exe T3.com
```

Wenn Sie keinen Assembler besitzen, können Sie dafür »t3.hex« (Listing 2) in das Utility SCHECK eingeben, daß sich ebenfalls in diesem Heft befindet. Nun kann der Aufruf von DOS-T3 in der Datei »autoexec.bat« installiert werden. Vergessen Sie nicht, den Aufruf des alten Treibers (Keybgr oder ähnliches) zu entfernen. Der Treiber wird mit »T3 0« aufgerufen. Für Besitzer einer Herculeskarte sind zusätzlich Parametereingaben zwischen 1 und 9 erlaubt.

(Karl-Heinz Schulz/ue)

```
[1V0] 0: .286C
[RX8] 1: TITLE T3 AT-Tastaturtreiber
[5E1] 2: NAME T3
[6S4] 3: SUBTTL Vorlauf
[S72] 4: PAGE 60,130
[V10] 5: ;
[688] 6: COMMENT $ AT-Tastaturtreiber
[9UC] 7: - Bildschirm-Tore (Vergl. DOS EXTRA 1, S
eite 127ff)
[3ZC] 8: Port 3D4H (Hercules 3B4H) CRT-6845 I
ndex-Register
[9ZC] 9: Port 3D5H (Hercules 3B5H) CRT-6845 D
aten-Register
[A6F] 10: Port 3D8H (Hercules 3B8H) Display Mo
de Control Port
[BCD] 11: Port 3DAH (Hercules 3BAH) Display St
atus Port
[5YP] 12: - Werden Interrupts über den 8259 erzeugt
, muß im Port 20H der Wert 20H (eoi)
[3DS] 13: ausgegeben werden, wenn wieder andere I
nterrupts zugelassen werden sollen.
[FPM] 14: Der erste Interrupteingang (IR0) ist f
ür den Zeitgeber zuständig
[B9M] 15: der zweite Interrupteingang (IR1) ist f
ür die Tastatur zuständig.
[LF7] 16: (Vergl DOS EXTRA 1, Seite 126)
[S8L] 17: - Der Zeitgeberinterrupt 1CH wird 18,2 ma
1 je Sekunde angesprungen
[BG8] 18: - Port 60H: Tastatur-Datentor
[82H] 19: Port 61H: Systemstatusbits (0-3: auch S
chreiben möglich)
[FMC] 20: 0 Timer2-Tor freigeben
4 Refresh-Tick
[76L] 21: 1 Lautsprecher-Tor freigeben
5 Timer2-Ausgang rücksetzen
[9RL] 22: 2 RAM-Parity-NMI sperren
6 AT:I/O-Kanal -Fehler/PC:Tastatur-Strobe
[5VM] 23: 3 I/O-Kanal NMI sperren
7 AT:RAM-Parity-Fehler/PC:Tastatur-Interrupt
[6H9] 24: Port 64H: Tastatur-Kommandotor
[1R0] 25: END $
[A56] 26: ;-----
[Y1A] 27: ;-- Konstantendeklaration
```

```
[A56] 28: ;-----
[HB4] 29: false EQU 0000H
[ON6] 30: true EQU not false
[AC4] 31: bell EQU 07H
[YU3] 32: LF EQU 0AH
[ZV3] 33: CR EQU 0DH
[SA8] 34: eoi EQU 20H ; 8259
EOI Wert
[OG4] 35: MSB EQU 10000000B
[585] 36: CmdLine EQU 80H
[M75] 37: ScrOn EQU 00001000B
[WYF] 38: Sekund EQU 3 ; Saver
-Basiseinheit für 30 Sec (1-120)
[A56] 39: ;-----
[Z28] 40: ;-- Port-Adressen
[A56] 41: ;-----
[9UF] 42: int_ctrl EQU 20H ; Inter
ruptkontrollport (8259)
[ACC] 43: Timer2 EQU 42H ; Lauts
precher (8254)
[7CN] 44: zeitgeber EQU 43H ; Steue
rregister für alle Timer (nur Schreiben)
[7AD] 45: kb_Data EQU 60H ; Tasta
tur-Datenport
[32D] 46: kb_Ctrl EQU 61H ; Syste
mstatusbits
[96F] 47: kb_Command EQU 64H ; Tasta
tur-Kommandostatus
[A56] 48: ;-----
[N47] 49: ;-- Interrupts
[A56] 50: ;-----
[92E] 51: Tast_Int EQU 09H ; IRQ1
Tastaturinterrupt
[NND] 52: Video EQU 10H ; Bilds
chirmservice 00..0F :
[EMB] 53: CurGr EQU 01H ; Cursorf
orm in DH,DL
[01G] 54: ScrGet EQU 0FH ; Modus h
olen: AL=Modus, AH=Spalten
[6AD] 55: Abbruch EQU 1BH ; CTRL-
Break-Behandlung
[AKD] 56: Time_Int EQU 1CH ; Zeitg
eber-Interrupt
```



```

[HUA] 57: Pend EQU 20H ; Progr
amm beenden
[ATB] 58: MsDos EQU 21H ; Funkt
ionsaufrufe
[23E] 59: CharOut equ 02H ; Zeichen
in DL ausgeben
[N2D] 60: StrgOut equ 09H ; Stringa
usgabe bis '$'
[AXL] 61: SetVec equ 25H ; Interru
pt-Vektoradresse setzen aus DS:DX
[T1L] 62: GetVec equ 35H ; Interru
pt-Vektoradresse holen in ES:BX
[S6G] 63: Terminate EQU 27H ; wie P
END, aber resident bleiben
[A56] 64: ;-----
[G59] 65: ;-- Ausgesuchte Scan-Kodes -
[A56] 66: ;-----
[9Y4] 67: released EQU 80H
[K75] 68: Sc_Extended EQU 00H
[V74] 69: Sc_Esc EQU 01H
[324] 70: Sc_Akzent EQU 0DH
[1T3] 71: Sc_BS EQU 0EH
[0A4] 72: Sc_Tab EQU 0FH
[PH3] 73: Sc_Z EQU 15H
[E25] 74: Sc_Return EQU 1CH
[HC4] 75: Sc_Ctl EQU 1DH
[HW4] 76: Sc_L_Shift EQU 2AH
[WL3] 77: Sc_Y EQU 2CH
[PV4] 78: Sc_R_Shift EQU 36H
[K25] 79: Sc_Prt_Scr EQU 37H
[XA4] 80: Sc_Alt EQU 38H
[8W4] 81: Sc_Caps EQU 3AH
[4M3] 82: Sc_F1 EQU 3BH
[2N3] 83: Sc_F10 EQU 44H
[S75] 84: Sc_Num_lock EQU 45H
[VY4] 85: Sc_Scroll EQU 46H
[R2C] 86: Sc_Home EQU 47H ; 7
im Nummernblock
[77B] 87: Sc_Up EQU 48H ; 8
im Nummernblock
[VWB] 88: Sc_Pg_Up EQU 49H ; 9
im Nummernblock
[KA5] 89: Sc_Gr_minus EQU 4AH
[25C] 90: Sc_Left EQU 4BH ; 4
im Nummernblock
[X4C] 91: Sc_Five EQU 4CH ; 5
im Nummernblock
[G4C] 92: Sc_Right EQU 4DH ; 6
im Nummernblock
[545] 93: Sc_Gr_plus EQU 4EH
[6LB] 94: Sc_End EQU 4FH ; 1
im Nummernblock
[4LB] 95: Sc_Dwn EQU 50H ; 2
im Nummernblock
[WTB] 96: Sc_Pg_Dwn EQU 51H ; 3
im Nummernblock
[BB4] 97: Sc_Ins EQU 52H
[584] 98: Sc_Del EQU 53H
[M35] 99: Sc_Sys_Req EQU 54H
[VN3] 100: Sc_F11 EQU 57H
[6P3] 101: Sc_F12 EQU 58H
[A56] 102: ;-----
[B58] 103: ;-- BIOS-Datensegment -
[YHA] 104: ;-- Einrichten der Offsetadressen -
[A56] 105: ;-----
[C46] 106: Bios_Data SEGMENT AT 40H
[4H5] 107: RS232_BASE DW 4 dup(?)
[336] 108: PRINTER_BASE DW 4 dup(?)
[SW5] 109: EQUIP_Flag DW ? ; 10H
[SME] 110: MFG_Flag DB ? ;
Initialisierungs-Flagge
[3X6] 111: Memory_Size DW ? ; 13H
[MMG] 112: MFG_err_Flag DB ? ;
zusätzliche Bytes zur Ablage
[TX9] 113: DB ? ;
von Fehler-Kodes
[A2E] 114: KB_FLAG DB ? ; BASIC> DEF SEG=
&H40: ? HEX$(PEEK(23)+(PEEK(24)*256)
[284] 115: INS_STATE EQU 80H
[PE4] 116: CAPS_STATE EQU 40H
[U84] 117: NUM_STATE EQU 20H
[TH4] 118: SCROL_STATE EQU 10H
[D84] 119: ALT_SHIFT EQU 8H
[Z74] 120: CTL_SHIFT EQU 4H
[TB4] 121: LEFT_SHIFT EQU 2H
[TH4] 122: RIGHT_SHIFT EQU 1H
[VGF] 123: KB_2_FLAG DB ? ; PASCAL> VAR peek
integer absolute $40:$17
[Y84] 124: INS_SHIFT EQU 80H
[KC4] 125: CAPS_SHIFT EQU 40H
[T84] 126: NUM_SHIFT EQU 20H

```

```

[KK4] 127: SCROL_SHIFT EQU 10H
[WM9] 128: HOLD_STATE EQU 8H ;
Ctrl-Num
[DXG] 129: Alt_Input DB ? ; 19H
Byte für ALT-Nummernblock-Eingabe
[WZL] 130: Buffer_Head DW ? ; 1AH
Zeiger auf Anfang des Tastaturpuffers
[GUL] 131: Buffer_Tail DW ? ; 1CH
Zeiger auf Ende des Tastaturpuffers
[XRM] 132: KB_Buffer DW 16 DUP(?) ; 1EH
Tastaturpuffer (16 Einträge zu je 2 Bytes)
[45D] 133: NEC_DISK DB 11 DUP(?) ;
Diskettenparameter
[VBF] 134: CRT_Mode DB ? ; 49H
aktueller Bildschirmmodus
[AYK] 135: CRT_Cols DW ? ;
Anzahl der dargestellten Spalten (40/80)
[5UH] 136: CRT_Len DW ? ;
Textzeilen pro Bildschirmseite
[EVM] 137: CRT_Start DW ? ;
Offsetadresse der aktuellen Bildschirmseite
[UWN] 138: Cursor_Posn DW 8 DUP(?) ;
Cursorkpositionen für 8 Bildschirmseiten
[XPL] 139: Cursor_Mode DW ? ; 60H
Cursorgöße in Rasterzeile Start/Ende
[YKF] 140: Active_Page DB ? ;
Aktuelle Bildschirmseite
[LZF] 141: ADDR_6845 DW ? ;
CRT Index-Register-Portadresse
[YS6] 142: CRT_Mode_Set DB ? ; 65H
[DY5] 143: CRT_Palette DB ? ;
[5CG] 144: IO_ROM_Init DD ? ; 67H
Pointer auf zusätzliches ROM
[MZH] 145: Intr_Flag DB ? ; 6BH
'Interrupt vorgekommen'-Flagge
[OS5] 146: Timer_Data DD ? ;
[Y7G] 147: DayFlag DB ? ;
Timer-Überlauf-(Tages-)Flagge
[277] 148: BiosBreakFlag DB ? ; 71H
[XA6] 149: Reset_Flag DW ? ; 72H
[4S7] 150: HardDisk_Var DW 2 DUP(?) ; 74H
[AMN] 151: Timeout_LPT DB 1,2,3,4 ;
Zeitzähler für 4 parallele Schnittstellen
[CDN] 152: Timeout_COM DB 1,2,3,4 ;
Zeitzähler für 4 serielle Schnittstellen
[AWG] 153: Buffer_Start DW ? ; 80H
zusätzliche Tastaturzeiger
[3D4] 154: Buffer_End DW ? ;
[EE6] 155: EGA_BIOS DB 4 DUP(?) ; 84H
[8V7] 156: Monitor_Typ DB 2 DUP(?) ; 88H
[BS6] 157: Grafik_Card DB ? ; 8AH
[BZ6] 158: Dummy_1 DB 5 DUP(?) ; 8BH
[KTG] 159: FloppyA_Flag DB ? ; 90H
ID-Byte für Disktyp Laufwerk A
[3UG] 160: FloppyB_Flag DB ? ; 91H
ID-Byte für Disktyp Laufwerk B
[74B] 161: Dummy_2 DB 5 DUP(?) ; 92H
reserviert
[AWG] 162: AT_Dioden DB ? ; 97H
Die Bits für die Tastaturdioden
[3L4] 163: CAPS_Diode EQU 04H
[HN4] 164: NUM_Diode EQU 02H
[ZL4] 165: SCROL_Diode EQU 01H
[DD1] 166: ORG 100H
[HKH] 167: HardCopyFlag DB ? ; 100H
(0:Hardcopy ok,1:busy,Offh:Error)
[RY3] 168: Bios_Data ENDS
[UA8] 169: ;=====
[NK8] 170: ;= Assembler: Segment- und =
[809] 171: ;= Organisationsbefehle =
[UA8] 172: ;=====
[4U4] 173: CSEG SEGMENT 'CODE'
[C6A] 174: ASSUME CS:CSEG, DS:Bios_Data
, SS:NOTHING
[GB2] 175: ORG 100H
[V10] 176: ;
[EHK] 177: Beginn: JMP Initialize ; I
nstalliere neuen INT9-Treiber
[026] 178: SUBTTL Datenbereich
[X21] 179: PAGE +
[UA8] 180: ;=====
[HWC] 181: ;= Residente Daten und Tastaturlisten=
[UA8] 182: ;=====
[SR0] 183: EVEN
[PA5] 184: old9H label dword
[W16] 185: old9H_Vector DW 2 DUP(?)
[DG5] 186: old1CH label dword
[N96] 187: old1CH_Vector DW 2 DUP(?)
[UB3] 188: IBM DB 'IBM'
[WCB] 189: ADAPTER DB ? ; 0
=MDA, 1=HGC, 2=CGA, 3=EGA

```



```

[BAD] 190: Video Port      DW      ?          ; 3
      D8H (3B8H für Hercules)
[N08] 191: CRTC_ADDR      DW      ?          ; 3
      D4H (3B4H)
[BB5] 192: VIDEO_SEGMENT  DW      0B800H
[5UB] 193: KENNUNG:       DB      '(c) 24.02.1988 KHS Kö
      ln, Tel. 0221-317232 '
[7Y4] 194: VERSION:      DB      'V3.11$'
[KF5] 195: Cursor_Modus   DW      ?
[G5B] 196: Orig_Flag      DB      false      ; (
      ungenutzt)
[ERE] 197: AKZENT        DB      false      ; i
      nterne Flagge für "'^"
[WCG] 198: Screen_Save   DB      false      ; T
      RUE==ScreenSave gewünscht
[37F] 199: Screen_Off    DB      false      ; T
      RUE==Schirm ist geblenkt
[KVD] 200: Screen_Tick   DW      ?          ; K
      onstante Wartezeit
[6PE] 201: Screen_Count  DW      ?          ; Z
      ähler für Wartezeit
[4R8] 202: TABLE:      DB      1BH, '1', '2', '3', '4', '5',
      '6', '7', ; 01H
[SD7] 203:              DB      '8', '9', '0', 225, 39, 08H,
      09H, 'q', ; 09H
[TK8] 204:              DB      'w', 'e', 'r', 't', 'z', 'u',
      'i', 'o', ; 11H
[XA8] 205:              DB      'p', 129, '+', 0DH, 04H, 'a',
      's', 'd', ; 19H
[A78] 206:              DB      'f', 'g', 'h', 'j', 'k', 'l',
      148, 132 ; 21H
[T58] 207:              DB      '<', 02H, '#', 'y', 'x', 'c',
      'v', 'b', ; 29H
[UD7] 208:              DB      'n', 'm', ' ', ' ', '-', 01H,
      '*', 08H ; 31H
[407] 209:              DB      ' ', 'e', ' ', '<', '=', '>',
      '?', 'e', ; 39H
[767] 210:              DB      'A', 'B', 'C', 'D', ' ', 10H,
      'G', 'H', ; 41H
[RK7] 211:              DB      'I', '-', 'K', 0FFH, 'M', '+',
      'O', 'P', ; 49H
[2V7] 212:              DB      'Q', 'R', 'S', '*', 0FFH, 0FFH,
      'W', 'X', ; 51H
[0FA] 213: TABL2:       DB      1BH, '!', '"', 15H, '$', '&',
      '&', '/', ; L-/R-SHIFT
[H36] 214:              DB      '(', ')', '=', '?', '\', 08H,
      00H, 'Q'
[LL6] 215:              DB      'W', 'E', 'R', 'T', 'Z', 'U',
      'I', 'O'
[4S6] 216:              DB      'P', 154, '*', 0DH, 04H, 'A',
      'S', 'D'
[9C6] 217:              DB      'F', 'G', 'H', 'J', 'K', 'L',
      153, 142 ; 00H, 08H
[GL6] 218:              DB      '>', 0FFH, '^', 'Y', 'X', 'C',
      'V', 'B'
[YC6] 219:              DB      'N', 'M', ' ', ' ', ':', '_', 01H,
      00H, 08H
[FK6] 220:              DB      ' ', 0FFH, 'T', 'U', 'V', 'W',
      'X', 'Y'
[GW6] 221:              DB      'Z', '[', '\', ']', 0FFH, 0FFH,
      '7', '8'
[SR5] 222:              DB      '9', '-', '4', '5', '6', ' ',
      '1', '2'
[ZL6] 223:              DB      '3', '0', ' ', '/', 0FFH, 0FFH,
      'w', 'x'
[SPA] 224: TABL3:       DB      0BH, 00H, '@', 0FFH, 0FFH, 0FFH,
      1EH, 0FFH ; CTRL
[138] 225:              DB      0FFH, 0FFH, 0FFH, 01FH, 0FFH, 7FH, 0
      FFH, 11H
[R17] 226:              DB      17H, 05H, 12H, 14H, 1AH, 15H,
      09H, 0FH
[NZ6] 227:              DB      10H, '{', '}', 0AH, 0FFH, 01H,
      13H, 04H
[TY6] 228:              DB      06H, 07H, 08H, 0AH, 0BH, 0CH,
      'e', 'e'
[187] 229:              DB      '|', 0FFH, 1CH, 19H, 18H, 03H,
      16H, 02H
[E78] 230:              DB      0EH, 0DH, 0FFH, 0FFH, 0FFH, 0FFH, 0
      FFH, 0FFH
[ES6] 231:              DB      ' ', 0FFH, '^', '_', ' ', 'a',
      'b', 'd'
[WE7] 232:              DB      'd', 'e', 'f', 'g', 0FFH, 0FFH,
      'w', 0FFH
[7A8] 233:              DB      84H, '-', 's', 0FFH, 't', '+',
      'u', 0FFH ; ä
[XU7] 234:              DB      'v', 0FFH, 0FFH, '/', 0FFH, 0FFH, 0
      FFH, 0FFH
[RFD] 235: TABL4:       DB      1BH, 0FFH, 0FFH, 0FFH, 0FFH, 0FFH, 0
      FFH, 0FFH ; ALT-Tabelle
[W38] 236:              DB      0FFH, 0FFH, 0FFH, 168, 174, 08H,
      09H, 0FFH ;

```

hp HEWLETT
PACKARD

EPSON

Wichmann AT 20 Student

Technische Daten:

512 kB Arbeitsspeicher, aufrüstbar
auf 1 MB · Taktrate 10 MHz, einstellbar
auf 8 MHz · keine Wartezyklen
(0-Waitstat) · Norton Faktor: > 11 · 8
Erweiterungsslots · lizenziertes BIOS
Uhr und Kalender akkugepuffert ·
80287 Co-Prozessor optional

Schnittstellen:

1 RS 232 (seriell) · 1 Centronics
(parallel)

Floppy Disk(FDU)- und Hard

Disk(HDU)-Controller:
Western Digital WD-1003, kombinier-
ter Controller für 2 FDU und 2 HDU

Floppy Disk:

1,2 MB NEC 1157 C slimline 5 1/4"
(optional 360 kB NEC 1054 slimline
5 1/4" oder 720 kB TEAC FD35FN
slimline 3,5")

Harddisk:

20 MB Seagate ST 225, mittlere
Zugriffszeit 65 ms

Netzteil:

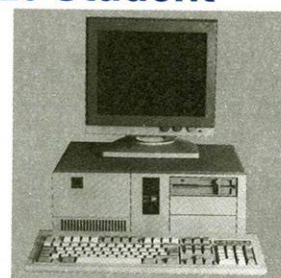
180 Watt Schaltnetzteil

Gehäuse:

Stahlblechgehäuse, lackiert, mit 3
slimline-Einschüben, Anzeigen für:
Taktrate/Betriebs-LED/HDU-
Symbol-LED/Schlüsselschalter

Tastatur:

Multifunktionsastatur mit ausga-
lertem Cursorblock, wie IBM MF 2.
Auf Wunsch andere Tastatur.



Bildschirmadapter:

monochrom, Hercules kompatibel

Monitor:

14" Samsung-Monitor bernstein,
entspiegelt, flatscreen, mit Dreh- und
Schwenkfuß optional schwarz/weiß

Besonderheiten:

Fertigung in Braunschweig ·
garantierter 24 h Service in eigener
Werkstatt · 12 Monate Garantie

3499,-
inkl. Mwst.



WICHMANN DATENTECHNIK GmbH

Bertramstr. 10 · 3300 Braunschweig
Telefon 05 31 / 7 25 83 u. 7 26 85
Telefax 05 31 / 7 69 61

Multitech Microsoft

PR8-SOFT

Telefon: 0931/464414

DER VIDEO DIGITISER VON ROMBO FÜR DEN PC

VIDEO DIGITISER

- Für PC XT/AT und Kompatible mit mindestens 256 K Haupt-
speicher und Hercules-, CGA- oder EGA-Karte (auch
SCHNEIDER/AMSTRAD PC 1512 monochrom)
- Anschluß beliebiger Videoquellen (COMPOSITE), z.B.
Videorecorder, Videokamera, TV etc.
- Komfortable Menüsteuerung über Cursortasten
- Definieren, Verschieben, Invertieren von Windows
- Auflösung in 16 Graustufen/Farben, Filterfunktion
- Speichern beliebiger Screens/Windows als GEM-Files
(2 Graustufen), Aldus-Pagemaker-Files (16 Graustufen) oder
VIDI-PC-Files
- Eigenes Grafik-Druckprogramm/EPSON-kompatibel

INFOS ANFORDERN!

Digitiser-Card zum Einstecken incl. Software,
Anschlußkabel und deutschem Handbuch

DM 448.-

Telefonische Eil-Bestellung (24 Stunden-Service): 0931/464414
PR8-SOFT Klaus-M. Pracht · Postfach 500 · D-8702 Margetshöchheim

Lieferung per Nachnahme (Versandkosten DM 5.- + NN-Gebühr) oder gegen Scheck
(+Versandk. DM 5.-). Auslandslieferungen gegen Scheck (+Versandkosten DM 10.-)

[AK9]	237:		DB OFFH,OFFH,OFFH,OFFH,OFFH,OFFH,OFFH,OFFH,OFFH
		FFH,OFFFH	; 11H
[HW8]	238:		DB OFFH, '[', ']', ODH, 04H,OFFH,0
		FFH,OFFFH	; 19H
[8C9]	239:		DB OFFH,OFFH,OFFH,OFFH,OFFH,OFFH,OFFH,OFFH,
		'{', '}'	; 21H
[1U8]	240:		DB '\', 02H, '#',OFFH,OFFH,OFFH,OFFH,0
		FFH,OFFFH	; 29H
[9R7]	241:		DB OFFH,OFFH, ',', ', .', '- , 01H,
		** , 08H	; 31H
[EP8]	242:		DB ' , ' , '@',OFFH,OFFH,OFFH,OFFH,OFFH,0
		FFH,OFFFH	; 39H
[VK8]	243:		DB OFFH,OFFH,OFFH,OFFH,OFFH, ' , 10H,
		'G' , 'H'	; 41H
[BT7]	244:		DB 'I' , 241, 'K',OFFH, 'M' , 241,
		'O' , 'P'	; 49H
[SC8]	245:		DB 'Q' , 'R' , 'S' , 246,OFFH,OFFH,0
		FFH,OFFFH	; 51H
[OBG]	246:	TABL A4:	DB 1BH,078H,079H,07AH,07BH,07CH,0
		7DH,07EH	; ALT-Tablelle Extended
[E18]	247:		DB 07FH,080H,081H,082H,083H, 08H,
		09H,010H	;
[EG8]	248:		DB 011H,012H,013H,014H,015H,016H,0
		17H,018H	; 11H
[OG8]	249:		DB 019H, '[' , ']' , ODH, 04H,01EH,0
		1FH,020H	; 19H
[EF8]	250:		DB 021H,022H,023H,024H,025H,026H,
		'{', '}'	; 21H
[TF8]	251:		DB '\', 02H, '#',02CH,02DH,02EH,0
		2FH,030H	; 29H
[AD7]	252:		DB 031H,032H, ',', ', .', '- , 01H,
		** , 08H	; 31H
[988]	253:		DB ' , ' , '@',068H,069H,06AH,06BH,0
		6CH,06DH	; 39H
[R68]	254:		DB 06EH,06FH,070H,071H, ' , 10H,
		'G' , 'H'	; 41H
[BT7]	255:		DB 'I' , 241, 'K',OFFH, 'M' , 241,
		'O' , 'P'	; 49H
[SC8]	256:		DB 'Q' , 'R' , 'S' , 246,OFFH,OFFH,0
		FFH,OFFFH	; 51H
[EAE]	257:	TABL S4:	DB 1BH, '1' , 171, '3' , 172, '5' , '
		6' , '7'	; ALT+SHIFT-Tabelle
[OX6]	258:		DB 236, '9' , 247, 225, 175, 08H, 0
		9H, 'q'	;
[508]	259:		DB 'w' , 144, 213, 209, 184, 'u' , 2
		45, 234	; 11H
[178]	260:		DB 14H, '[' , ']' , ODH, 04H, 224, 2
		28, 7FH	; 19H
[CN7]	261:		DB 198, 216, 181, 'j' , 'k' , 156, '
		@' , '@'	; 21H
[HH7]	262:		DB 126, 02H, '#' , 157, 'x' , 128, 2
		12, 207	; 29H
[3C7]	263:		DB 190, 'm' , ',', ', .', '- , 01H, '
		** , 08H	; 31H
[G27]	264:		DB ' , ' , '@' , ',' , '<' , '=' , '>' , '
		? , '@'	; 39H
[ZE7]	265:		DB 'A' , 'B' , 'C' , 'D' , ' , 10H, '
		G' , 'H'	; 41H
[FV7]	266:		DB 'I' , 241, 'K',OFFH, 'M' , 241, '
		O' , 'P'	; 49H
[ZS8]	267:		DB 'Q' , 'R' , 'S' , 246,OFFH,OFFH,OF
		FH,OFFFH	; 51H
[EZC]	268:	TABL5:	DB 1BH, '1' , 253, '3' , 172, '5' , '
		6' , '7'	; ALT-CTRL-Tabelle
[UT6]	269:		DB '8' , '9' , 243, 225, 169, 08H, 0
		9H, 'q'	;
[KW7]	270:		DB 'w' , 138, 218, 194, 191, 196, '
		i' , 167	; 11H
[NB8]	271:		DB 232, '{' , '}' , ODH, 04H, 166, '
		s' , 'd'	; 19H
[9N7]	272:		DB 195, 197, 180, 179, 'k' , 156, '
		[' , ']'	; 21H
[9X7]	273:		DB ' ' , 02H, '#' , 157, 'x' , 'c' , 1
		92, 193	; 29H
[3M7]	274:		DB 217, 'm' , ',', ', .', '- , 01H, '
		** ,OFFFH	; 31H
[G27]	275:		DB ' , ' , '@' , ',' , '<' , '=' , '>' , '
		? , '@'	; 39H
[ZE7]	276:		DB 'A' , 'B' , 'C' , 'D' , ' , 10H, '
		G' , 'H'	; 41H
[XC8]	277:		DB 'I' ,OFFFH, 'K',OFFFH, 'M',OFFFH, '
		O' , 'P'	; 49H
[ZS8]	278:		DB 'Q' , 'R' , 'S' , 246,OFFH,OFFH,OF
		FH,OFFFH	; 51H
[AFE]	279:	TABL S5:	DB 1BH, '1' , 253, '3' , 172, '5' , '
		6' , '7'	; ALT+CTRL+SHIFT-Tabelle
[9T6]	280:		DB '8' , '9' , 242, 225, 170, 08H, 0
		9H, 'q'	;
[H18]	281:		DB 'w' , 138, 214, 210, 183, 'u' , '
		i' , 167	; 11H

```

[NB8] 282: DB 232, '{', '}', 0DH, 04H, 166, '
      s', 'd' ; 19H
[OS7] 283: DB 199, 215, 182, 'j', 'k', 156, '
      [, ']' ; 21H
[VM7] 284: DB '|', 02H, '#', 157, 'x', 'c', 2
      11, 208 ; 29H
[EN7] 285: DB 189, 'm', ',', '.', '-', 01H, '
      *, OFFH ; 31H
[G27] 286: DB ', ', '@', ', ', '<', '=', '>', '
      ?, '@' ; 39H
[ZE7] 287: DB 'A', 'B', 'C', 'D', ' ', 10H, '
      G', 'H' ; 41H
[XC8] 288: DB 'I', OFFH, 'K', OFFH, 'M', OFFH, '
      O', 'P' ; 49H
[ZS8] 289: DB 'Q', 'R', 'S', 246, OFFH, OFFH, OF
      FH, OFFH ; 51H
[32E] 290: TABL X: DB OFFH, 'x', 'y', 'z', '{', '|',
      '}', '~', ; extended Kode
[MBA] 291: DB 7FH, 80H, 81H, 82H, 83H, OFFH, 0
      FFH, 10H ; Cüéá
[8Z6] 292: DB 11H, 12H, 13H, 14H, 15H, 16H,
      17H, 18H
[1R7] 293: DB 19H, OFFH, OFFH, OFFH, OFFH, 1EH,
      1FH, ' '
[2Z5] 294: DB '!', '"', '#', '$', '%', '&', 0
      FFH, OFFH
[4U6] 295: DB OFFH, OFFH, OFFH, ' ', '-', '.', '
      ', '/', '0'
[PM7] 296: DB '1', '2', OFFH, OFFH, OFFH, OFFH, 0
      FFH, OFFH
[BY6] 297: DB ' ', OFFH, 'h', 'i', 'j', 'k',
      'l', 'm'
[6P7] 298: DB 'n', 'o', 'p', 'q', OFFH, OFFH, 0
      FFH, OFFH
[4F8] 299: DB OFFH, OFFH, OFFH, OFFH, OFFH, OFFH, 0
      FFH, OFFH
[4F8] 300: DB OFFH, OFFH, OFFH, OFFH, OFFH, OFFH, 0
      FFH, OFFH
[K5D] 301: Akz Tabl: DB 027H, '^', '^'
      ; Akzenttabelle
[7MA] 302: DB 091H, 091H, 086H, 092H, 092H, 0
      8FH ; ä - ääääA
[3MA] 303: DB 0A0H, 085H, 083H, 086H, 0A6H, 0
      8FH ; a - ääääaA
[CBA] 304: DB 087H, 087H, 087H, 080H, 080H, 0
      80H ; c - cccccc
[8GA] 305: DB 082H, 08AH, 088H, 090H, 090H, 0
      90H ; e - ééééE
[DA8] 306: DB 0A1H, 08DH, 08CH
      ; i - iiii
[D9B] 307: DB 0A4H, 0A4H, 0A4H, 0A5H, 0A5H, 0
      A5H ; n - nnnnnN
[HA8] 308: DB 0A2H, 095H, 093H
      ; o - ööö
[216] 309: DB 0A3H, 097H, 096H
      ; u -
[H88] 310: DB 089H, 08BH, 098H
      ; y - üüü
[LN5] 311: SUBTTL ScreenSaver
[X21] 312: PAGE +
[UA8] 313: ;=====
[6UA] 314: ;= Ab zur alten (BIOS-)Timerroutine =
[UA8] 315: ;=====
[TV3] 316: Timer PROC NEAR
[2X9] 317: POPA ; (
      nur bei AT möglich)
[1D1] 318: POP ES
[UC1] 319: POP DS
[CB2] 320: JMP old1CH
[F03] 321: Timer ENDP
[UA8] 322: ;=====
[54B] 323: ;= Hier ist jetzt die neue INT1CH-Ein- =
[GLC] 324: ;= sprungs-/Verarbeitungsadresse =
[UA8] 325: ;=====
[VG3] 326: NEW1CH PROC NEAR
[FK1] 327: PUSH DS
[PK1] 328: PUSH ES
[S3A] 329: PUSHA ; (
      nur bei AT möglich)
[9D4] 330: MOV AX, Bios_Data
[VV1] 331: MOV DS, AX
[51F] 332: DEC CS: Screen_Count ; Z
      ähler dekrementieren
[E72] 333: JNZ Timer
[GLD] 334: CMP CS: Screen_Off, true ; I
      F Schirm schon aus ?
[EHD] 335: JZ R_Count ; T
      HEN nur Zähler rücksetzen
[NAC] 336: MOV CS: Screen_Off, true ; E
      LSE ausschalten

```

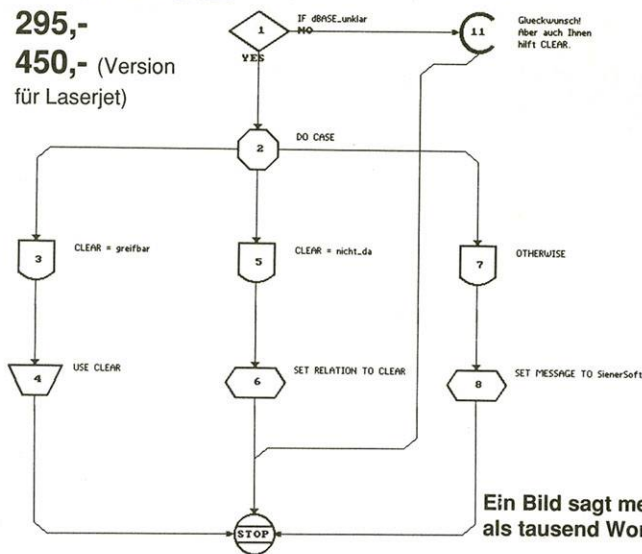

US SOFTWARE IMPORT SERVICE

Alle Produkte werden als original US-Software-Lizenzen geliefert.

CLEAR™

for dBASE

295,-
450,- (Version
für Laserjet)



Ein Bild sagt mehr
als tausend Worte.

Daß eine geordnete Dokumentation notwendig ist, sieht jeder Programmierer ein. Wenn's aber schnell gehen muß?

Nicht nur dBASE-Benutzer können dieses leidige Problem jetzt vergessen. **CLEAR** macht aus Ihrem Quellcode automatisch eine komplette Dokumentation: Baumstrukturen, Flußdiagramme und hervorragend gestaltete Listings. Ab etwa August '88 wird der erweiterte Befehlssatz von Foxbase, Clipper und Quicksilver Diamond von Wordtech voll unterstützt. Auf allen Ausdrucken können viele wichtige Angaben und Kommentare vermerkt werden.

CLEAR sollte aber nicht nur zur nachträglichen Dokumentierung eingesetzt werden. Es eignet sich bestens als Entwicklungsoberfläche. Ohne Schwierigkeiten erreichen Sie Ihren "Lieblingseditor", Ihren Datenbankinterpreter (-compiler) und weitere Programme. Eine Programmablauf-Simulation im Flußdiagramm ist möglich.

MASTER*KEY Disassembler - Version 2.2 - vollständig überarbeitet -

195,-

Intelligenter MS-DOS Disassembler für die Intel-Prozessoren 8088, 8086, 80186 und 80286 sowie 8087 und 80287.

MASTER*KEY erstellt aus Objekt-Files oder ausführbaren Programmen übersichtliche, dokumentierte Assembler-Quelldateien, die mit einem beliebigen ASCII-Textprogramm bearbeitet und anschließend neu assembliert und ggf. mit anderen Moduln gelinkt werden können. **MASTER*KEY** arbeitet unter MS-DOS 2.0, 2.2, 3.0, 3.1 und 3.2.

Für den 8086 enthält **MASTER*KEY** außerdem einen Code-Formatierer und einen Crossreference-Generator. Der Formater ist in der Lage, alle ROM-BIOS- und DOS-Funktionen und -Interrupts im erzeugten Assembler-Listing auszuweisen. Indem er hilft, Stack-Bereiche, Temporärspeicher, ASCII-Strings und überflüssigen Code besser zu erkennen, dient er gewissermaßen als Objektcode-Optimizer.

CROSS-16 Meta-Assembler

342,-

Anders als ein üblicher Crossassembler ist **CROSS-16** in der Lage, Programme für eine Vielzahl von Ziel-Prozessoren zu verarbeiten. Er liest eine Quelldatei und die passende Assembler-Befehlstabelle und gibt ein assembliertes Listing und eine Datei in Maschinensprache im Hex-Format aus. Weil er mit flexiblen Befehlstabellen arbeitet, eignet sich der **CROSS-16** für die meisten Mikroprozessoren und -controller, die mit einem 20-Bit-Adreßbus (oder weniger) arbeiten, insbesondere Intel- und Motorola-8- bzw. 16-Bit-Formate.

Btrieve - Record Management System

Mit **Btrieve** schreiben Sie Ihre Programme leichter und schneller, weil Sie sich um physische Dateistrukturen nicht mehr zu kümmern brauchen. Beschränken Sie sich auf die Logik! Schnittstellen für viele populäre Programmiersprachen, wie IBM-, Microsoft-BASIC, Turbo-Pascal und Microsoft C sind enthalten.

Mit **Btrieve** können Sie bis zu 24 Schlüsselfelder definieren, wobei für jeden Zugriffspfad ein eigener Index geführt wird. Die Index-Dateien werden automatisch aktualisiert.

Vier Indexattribute - duplicate, modifiable, segmented und null key - können vergeben werden. Daneben stehen alle Möglichkeiten einer relationalen Datenbank zur Verfügung. Bei einer max. Dateigröße von 4 Gigabyte können Files auf 2 Platten verteilt werden. Cache-Speicher und Integritätskontrollen zählen zu den weiteren Features.

Im Multi-User-Betrieb bzw. im lokalen Netzwerk ermöglicht **Btrieve** Simultanzugriff und konkurrierendes Update. Wirksamer Datenschutz durch Passwortvergabe mit Option "read-only" wird ebenfalls unterstützt.

Btrieve 590,-
Btrieve/Net 1.195,-

IMPORT-SERVICE:

Bitte rufen Sie uns an, wenn Sie ein Programm nicht in unserer Preisliste finden. Wir holen die neuesten Programme aus den USA.

LIEFERBEDINGUNGEN:

Alle Preise enthalten Versandkosten. Bei Zahlung per Vorkasse können Sie je Produkt DM 4,- vom Preis abziehen. Preisänderungen und Irrtum vorbehalten.

Bücher	
Using Network	85
Inside OS/2	99
The MS-DOS Encyclopedia	250

Datenbanken	
Paradox 2.0	1.315
Dataflex multi-user	2.490
Dataflex Runtime, multi	575
Dataflex Runtime, single	399
Dataflex single user	1.350
dB Fast, dBase Compiler	295
Foxbase+ 2.0	675
Foxbase+ 2.0 Multi-User	1.149
Clipper, Summer 87	1.295

Expertensysteme	
Arity Prolog Interpreter	997
GoldWorks	3.176
VP Expert	269
THE RESIDENT EXPERT System	269

Kommunikation	
Carbon Copy plus	395
Crosstalk Mark IV	385
Smarternet 240	695
Lap Link Mac	349
Lap Link plus	349
The Brooklyn Bridge	349

Pascal	
Turbo Pascal 4.0	199
Turbo Pascal Database 4.0	219
Turbo Pascal Editor 4.0	219
Turbo Pascal Graphix 4.0	219
Turbo Pascal Numerical Methods 4.0	219
Advantage 386 Pascal	2.865
Science & Engineering Tools	299
Overlay Manager 4.0	149
TDebugPLUS	99
TDebugPLUS with Source	149
Turbo Analyst 4.0	169
Turbo Professional 4.0	199

Programmiersprachen	
Macro Assembler 5.1 m. OS/2 Support	375
Turbo Basic 1.1	199
Turbo C 1.5	199

Microsoft C 5.1 Compiler mit OS/2-Support

795,-

C. talk, object oriented C	531
MS-Fortran Compiler 4.1	895
Quick C	199
QuickBASIC 4.0	199
Smalltalk/V	342

Utilities	
Fanci Console	249
Precursor	185
Tree 86	134
Keyworks Advanced	545
Pizzazz	105
ORG Plus	300
Sidekick plus	385
Memory Mate	162
Copy II PC	75
Copy II PC Option Board deluxe	375
PC Tools de Luxe	149
SPF/PC 2.0, deutsch	684
SPF/PC 2.0, US	513
XTree	119
XTree PRO	199
Fastback plus	359
QDOS II	149
Interactive EasyFlow	443
LaserPlotter	285
Matrix Plotter (Epson o. Toshiba)	399
Advanced Hard Disk Diag. 2.0	189
LIMSIM LIM 4.0 Simulation	295
Tornado Notes	295
Mace Utilities 4.1	219
Norton Commander	149
Norton Editor	282
Norton Guides Assembly	165
Norton Guides Basic	165
Norton Guides C	165
Norton Guides OS/2	185
Norton Utilities Advanced Edition	285
Corporate PVCs	969
Disk Technician+	349
PrintQ	199

Windows	
Windows 2.03	295
Windows 2.03 Development Toolkit	1.262
Windows 386	475
Desqview 2.0	295
Desqview Companions	212
QEMM 386	149
QEMM 50/60	149

Siener Soft

Ein Service der Firma:
Roland Siener Computerhandel
Gustav-Freytag-Straße 10
6200 Wiesbaden
Telefax 06121/309329
06121/309041


```

[K64] 337:      MOV     DX,CS:CRTC_ADR
[XKA] 338:      MOV     AL,6                ; B
        ildschirmregister 6
[VRB] 339:      OUT     DX,AL                ;
        Anzahl der sichtbaren
[MK7] 340:      INC     DX                ;
        Zeilen auf
[AW8] 341:      XOR     AX,AX                ;
        Null setzen
[SU1] 342:      OUT     DX,AL
[YY8] 343:      R_Count: MOV     AX,CS:Screen_Tick
[4E6] 344:      MOV     CS:Screen_Count,AX
[HRA] 345:      JMP     Timer                ; z
        um alten Interrupt
[MN2] 346:      NEWICH ENDP
[CH7] 347:      SUBTTL  Tastaturtreiber
[X21] 348:      PAGE +
[UA8] 349:      ;=====
[RWB] 350:      ;= Ab zum alten (BIOS-)Tastaturtreiber =
[UA8] 351:      ;=====
[SU3] 352:      Ext_9H  PROC  NEAR
[2X9] 353:      POPA                ; (
        nur bei AT möglich)
[1D1] 354:      POP     ES
[UC1] 355:      POP     DS
[B32] 356:      JMP     old9H
[EZ2] 357:      Ext_9H  ENDP
[UA8] 358:      ;=====
[UA8] 359:      ;= Bildschirm wenn geblinkt wieder =
[LBZ] 360:      ;= einschalten mit spezieller Abfrage =
[UA8] 361:      ;=====
[51B] 362:      Scr_On: MOV     DX,CS:CRTC_ADR ; (3x4H+6H
        )=3xÄH
[GOA] 363:      ADD     DX,6                ; Display
        Status Port
[PL3] 364:      Scr1:  IN     AL,DX
[5AB] 365:      TEST    AL,MSB                ; warte au
        f vertikales
[AKC] 366:      JZ      Scr1                ; Synchron
        isationssignal aus
[ZUC] 367:      Scr2:  IN     AL,DX                ; --- also
        Anfang des Signals
[5AB] 368:      TEST    AL,MSB                ; warte au
        f vertikales
[LNC] 369:      JNZ     Scr2                ; Synchron
        isationssignal ein
[MC4] 370:      INC     DX                ; 3xBH
[3U1] 371:      XOR     AL,AL
[SDD] 372:      OUT     DX,AL                ; lösche L
        ichtgriffel-Flipflop
[W57] 373:      INC     DX                ; 3xCH wen
        n xDxH
[NC2] 374:      MOV     AX,0B0H
[785] 375:      AND     AX,DX                ; Test
[F48] 376:      CMP     AL,0B0H                ; IF NOT H
        ercules
[EC6] 377:      JNE     SET_LT                ; THEN J
        UMP
[APC] 378:      SUB     DX,03H                ; ELSE A
        dresse 3B9H von Hercules
[2NF] 379:      SET_LT: OUT     DX,AL                ; setze Li
        chtgriffel-Flipflop
[C52] 380:      MOV     AL,10h
[PA9] 381:      MOV     DX,CS:CRTC_ADR ; lese Adr
        esse,
[Y06] 382:      OUT     DX,AL                ; bei de
        r
[Y1A] 383:      INC     DX                ; Lichtg
        riffelimpuls
[6A8] 384:      IN     AL,DX                ; auftra
        t, nach BX
[F51] 385:      DEC     DX
[0U1] 386:      MOV     BH,AL
[G52] 387:      MOV     AL,11h
[SU1] 388:      OUT     DX,AL
[661] 389:      INC     DX
[NP1] 390:      IN     AL,DX
[F51] 391:      DEC     DX
[WS8] 392:      MOV     BL,AL                ; vergleic
        he diese
[TN9] 393:      CMP     BX,0B8Dh                ; mit
        dem Wert 0b8dh
[0T8] 394:      MOV     AH,25                ; if BX <
        b8d # Zeilen
[2F2] 395:      JC      Sc_Set
[GZ8] 396:      MOV     AH,87                ; if BX
        b8d # Zeilen
[D0D] 397:      Sc_Set: MOV     AL,06                ; Schreibe
        Wert in AH nach
[SSD] 398:      OUT     DX,AL                ; Register

```

```

        6 des Videocontrollers
[7ZA] 399:      MOV     AL,AH                ; # der si
        chtbaren Zeilen
[661] 400:      INC     DX
[SU1] 401:      OUT     DX,AL
[SE0] 402:      RET
[UA8] 403:      ;=====
[NE4] 404:      ;=
[UA8] 405:      ;=====
[6N2] 406:      EXT_AH: CLI
[MC5] 407:      OR      [AT_Dioden],AH
[L75] 408:      RDY_15: JMP     FN_FERTG
[UA8] 409:      ;=====
[G4B] 410:      ;= Controll-Port-Steuerung mit =
[A89] 411:      ;= Interrupt 15H-Behandlung =
[UA8] 412:      ;=====
[MHA] 413:      INT15_NG:TEST    BYTE PTR [KB_2_FLAG],0000
        0100B
[W03] 414:      JNZ     RDY_15
[MX7] 415:      OR      BYTE PTR [KB_2_FLAG],0000
        0100B
[FF2] 416:      XOR     AL,AL
[LD2] 417:      JMP     INT15
[V10] 418:      ;
[R8A] 419:      INT15_RL:AND     BYTE PTR [KB_2_FLAG],1111
        1011B
[HR2] 420:      MOV     AL,01H
[4HG] 421:      INT15: MOV     AH,85H                ; T
        est auf System-Request-Taste
[XR1] 422:      PUSH    AX
[P4C] 423:      CALL    End_Int                ; I
        nterrupt freigeben
[ZCE] 424:      CALL    Freigabe                ; T
        astatur wieder freigeben
[8K1] 425:      POP     AX
[PE0] 426:      STI
[2R1] 427:      INT     15H
[253] 428:      JMP     FERTIG
[UA8] 429:      ;=====
[ZXA] 430:      ;= Hier ist jetzt die neue INT9H-Ein- =
[G1C] 431:      ;= sprungs-/Verarbeitungsadresse =
[UA8] 432:      ;=====
[BA2] 433:      NEW9H:  STI
[7D0] 434:      CLD
[FK1] 435:      PUSH    DS
[PK1] 436:      PUSH    ES
[S3A] 437:      PUSHA                ; (
        nur bei AT möglich)
[664] 438:      MOV     AX,Bios_Data
[ZU1] 439:      MOV     DS,AX
[N3E] 440:      CALL    Sperren                ; T
        astatur erstmal sperren
[V10] 441:      ;
[ZYG] 442:      CMP     CS:Screen_Off,false                ; I
        F Bildschirm eingeschaltet
[5DG] 443:      JE      ResCount                ; T
        HEN weiter mit Zähler rücksetzen
[ACD] 444:      CALL    Scr_On                ; E
        LSE Bildschirm einschalten
[6P7] 445:      MOV     CS:Screen_Off,false
[R09] 446:      ResCount:MOV    AX,CS:Screen_Tick
[RG6] 447:      MOV     CS:Screen_Count,AX
[V10] 448:      ;
[5PA] 449:      IN     AL,kb_Data                ; H
        ole den Scan-Kode
[KY1] 450:      MOV     AH,10H
[Y92] 451:      CMP     AL,0FAH
[P02] 452:      JZ      EXT_AH
[2S5] 453:      ADD     AH,AH                ; A
        H=20H
[EA2] 454:      CMP     AL,0FEH
[P02] 455:      JZ      EXT_AH
[V10] 456:      ;
[RY1] 457:      CMP     AL,69H
[MS2] 458:      JE      INT15_NG
[515] 459:      CMP     AL,69H+released
[6U2] 460:      JE      INT15_RL
[V10] 461:      ;
[TGH] 462:      TEST    AL,released                ; D
        auergedrückte Taste losgelassen
[KNC] 463:      JZ      TASTE_NG                ; n
        ein, neue Taste gedrückt
[V10] 464:      ;
[VR3] 465:      CALL    FUNC_TEST
[YAA] 466:      JC      RL_FUNC                ; C
        RY:=1 -> BL:= xlat (al-1)
[EZ2] 467:      JMP     FN_FERTG
[V10] 468:      ;
[0V5] 469:      TASTE_NG:CALL    FUNC_TEST
[2AA] 470:      JC      NG_FUNC                ; C

```



```

RY:=1 -> BL:= xlat (al-1)
[E13] 471:      JMP     NG_TASTE
[UA8] 472:      ;=====
[DP3] 473: RL_FUNC: PUSH    DI
[ME6] 474:      MOV     DI,offset KB_2_FLAG
[VE2] 475:      TEST    BL,0F0H
[OH3] 476:      JNZ     RL2_Func
[X96] 477:      MOV     DI,offset KB_FLAG
[L65] 478: RL2_Func: XOR     BL,0FFH
[HG2] 479:      AND     [DI],BL
[UC1] 480:      POP     DI
[423] 481:      CMP     AL,Sc_Alt
[1C2] 482:      JNZ     Fn_End
[9U1] 483:      XOR     AX,AX
[GW4] 484:      MOV     AL,[Alt_Input]
[LL7] 485:      MOV     BYTE PTR [Alt_Input],AH ; =
0
[CM1] 486:      OR      AL,AL
[H2C] 487:      JZ      Fn_End ; k
eine Eingabe im Speicher
[2ZD] 488:      JMP     ALT_NORM ; s
onst ALT-# Eingabe vorhanden
[V10] 489: ;
[PV4] 490: NG_FUNC: TEST    BL,0F0H
[KD1] 491:      JZ      B3F0
[724] 492:      TEST    BL,NUM_STATE
[DR2] 493:      JNZ     NUM_TST
[AG4] 494:      TEST    BL,SCROL_STATE
[783] 495:      JNZ     SCRL_TST
[AM6] 496: NGF_JP:  TEST    BL,[KB_2_FLAG]
[1C2] 497:      JNZ     Fn_End
[U24] 498:      OR      [KB_2_FLAG],BL
[ZW3] 499:      XOR     [KB_FLAG],BL
[90A] 500:      TEST    BL,MSB ; I
F Funktionstaste
[C99] 501:      JZ      SetDiode ; T
HEN SetDiode
[PU9] 502:      JMP     NG_TASTE ; E
LSE Normaltaste
[V10] 503: ;
[VR4] 504: B3F0:  OR      [KB_FLAG],BL
[RR5] 505: Fn_End: JMP     FN_FERTG
[V10] 506: ;
[MHA] 507: NUM_TST: TEST    BYTE PTR [KB_FLAG],CTL_SHIFT
[242] 508:      JZ      NGF_JP
[EX1] 509:      JMP     PAUSE
[V10] 510: ;
[8HA] 511: SCRL_TST:TEST    BYTE PTR [KB_FLAG],CTL_SHIFT
[242] 512:      JZ      NGF_JP
[AT1] 513:      JMP     BREAK
[1L5] 514: ;-----
[RF9] 515: ;- DATA-SEG: AT_Dioden-Behandlung -
[1L5] 516: ;-----
[MD3] 517: SetDiode:CLI
[UH1] 518:      PUSH    AX
[8K1] 519:      PUSH    CX
[5A5] 520:      TEST    [AT_Dioden],40H
[6K1] 521:      JNZ     B_B6
[LR3] 522:      MOV     AL,[KB_FLAG]
[91A] 523:      RCR     AL,4 ; (
nur bei AT möglich)
[1M1] 524:      AND     AL,07
[MT1] 525:      MOV     CL,AL
[6P4] 526:      XOR     AL,[AT_Dioden]
[1M1] 527:      AND     AL,07
[UD1] 528:      JZ      B_B6
[PD0] 529:      CLI
[LF5] 530:      AND     [AT_Dioden],0F8H
[6T4] 531:      OR      [AT_Dioden],CL
[H24] 532:      OR      [AT_Dioden],40H
[VSB] 533:      CALL    End_Int ; I
nterrupt freigeben
[7G4] 534:      MOV     AL,0EDH
[PN1] 535:      CALL    B_C6 ;
[PD0] 536:      CLI
[HA5] 537:      TEST    [AT_Dioden],80H
[4K1] 538:      JNZ     B_B0
[AT1] 539:      MOV     AL,CL
[PN1] 540:      CALL    B_C6
[PD0] 541:      CLI
[HA5] 542:      TEST    [AT_Dioden],80H
[UD1] 543:      JZ      B_B6
[2X5] 544: B_B0:  MOV     AL,0F4H ;
[PN1] 545:      CALL    B_C6
[PD0] 546:      CLI
[VH6] 547: B_B6:  AND     [AT_Dioden],3FH
[NS2] 548: B_BB:  POP     CX
[5C1] 549:      POP     AX
[PE0] 550:      STI
[E22] 551:      JMP     FN_FERTG
[V10] 552: ;

```

```

[EY2] 553: B_C6:  PUSH    AX
[8K1] 554:      PUSH    CX
[FK1] 555:      PUSH    DX
[6T1] 556:      MOV     AH,AL
[MP1] 557:      MOV     DL,05
[D22] 558: B_CE:  CLI
[XE4] 559:      IN      AL,kb_Command
[G94] 560:      TEST    AL,02 ;
[FL1] 561:      JNZ     B_CE
[US1] 562:      MOV     AL,AH
[GU3] 563:      OUT     kb_Data,AL
[RW5] 564:      AND     [AT_Dioden],01001111B
[PK2] 565:      MOV     CX,1B00H
[PE0] 566:      STI
[5S6] 567: B_E1:  TEST    [AT_Dioden],10H
[083] 568:      JNZ     B_C6_End
[Z95] 569:      TEST    [AT_Dioden],20H
[CK1] 570:      JNZ     B_F2
[CR1] 571:      LOOP    B_E1
[NB6] 572: B_F2:  OR      [AT_Dioden],80H
[LA1] 573:      DEC     DL
[FL1] 574:      JNZ     B_CE
[VN3] 575: B_C6_End:POP    DX
[LC1] 576:      POP     CX
[5C1] 577:      POP     AX
[SE0] 578:      RET
[UA8] 579: ;=====
[NY8] 580: ;= STI - Interrupt enable =
[UA8] 581: ;=====
[BYH] 582: End_Int: MOV     AL,EOI ; D
amit sind weitere Interrupts
[97H] 583:      OUT     int_ctrl,al ; w
ieder zulässig, bis jetzt wurde
[0F8] 584:      RET ; S
TI ignoriert!
[UA8] 585: ;=====
[CHB] 586: ;= Schreibe [AL,AH] in Tastaturpuffer =
[67B] 587: ;= Bei Pufferüberlauf erfolgt Signal =
[UA8] 588: ;=====
[ZT8] 589: PUSH_IT: MOV     BX,[Buffer_Tail]
[AY1] 590:      MOV     SI,BX
[WAL] 591:      INC     BX
[WAL] 592:      INC     BX
[ZRC] 593:      CMP     BX,[Buffer_End] ; I
F Inhalt identisch
[SFA] 594:      JNE     PUSH_1 ; T
HEN keine Operation
[7AH] 595:      MOV     BX,[Buffer_Start] ; E
LSE lade für zweiten Vergleich
[7TF] 596: PUSH_1: CMP     BX,[Buffer_Head] ; I
F Inhalt identisch
[1WG] 597:      JZ      OVERFLOW ; T
HEN Pufferüberlauf (Rückkehr von dort)
[3TB] 598:      MOV     [SI],AX ; E
LSE schreibe in Puffer
[T4K] 599:      MOV     [Buffer_Tail],BX ;
und inkrementiere Pufferzeiger
[V6A] 600:      CLC ;
lösche Fehlerflag
[ZHD] 601:      RET ;
zurück zum Aufruf von PUSH_IT
[V10] 602: ;
[2FE] 603: OVERFLOW:CALL    End_Int ; I
nterrupt freigeben
[3AG] 604:      MOV     AL,0B6h ; Z
eitgeber-Modus-Register schreiben
[4W4] 605:      OUT     zeitgeber,AL
[UH1] 606:      PUSH    AX
[5C1] 607:      POP     AX
[WG2] 608:      MOV     AX,0400H
[19F] 609:      OUT     Timer2,AL ; Z
ähler in Low-Byte Zeitgeber 2
[BW1] 610:      XCHG    AH,AL
[TDF] 611:      OUT     Timer2,AL ; Z
ähler in High-Byte Zeitgeber 2
[N9E] 612:      IN      AL,KB_Ctrl ; a
ktuelle Einstellung holen
[B07] 613:      PUSH    AX ; s
ichern
[CB3] 614:      OR      AL,03 ; L
autsprecher einschalten
[PP3] 615:      OUT     KB_Ctrl,AL
[BD7] 616:      MOV     CX,8000H ; w
arten
[U44] 617: here1: LOOP    here1
[NE9] 618:      POP     AX ; a
lten Inhalt holen
[9HG] 619:      OUT     KB_Ctrl,AL ; u
nd damit Lautsprecher abschalten
[86E] 620:      CALL    Freigabe ; T

```



```

[ER9] 621:          astatur wieder freigeben
          STC
[ZHD] 622:          setze Fehlerflag
          RET
          zurück zum Aufruf von PUSH_IT
[UA8] 623: ;=====
[55C] 624: ;= Treiberrücksprung (Interrupt-Ende) =
[ZW7] 625: ;= COD_EXT : SCAN und ZEICHEN =
[578] 626: ;= COD_NORM : ZEICHEN und SCAN =
[UA8] 627: ;=====
[DF3] 628: COD_EXT: DEC AL
[VZ5] 629:          XLAT Byte PTR CS:Table
[A78] 630:          CMP AL,OFFH ; K
          einen Kode
[SV2] 631:          JZ FN_FERTG
[GTA] 632:          MOV AH,AL ; S
          can-Kode (erweitert)
[Z2B] 633:          XOR AL,AL ; K
          ennung erweitert (0)
[V03] 634:          JMP ALT_NORM
[V10] 635: ;
[D89] 636: COD_NORM:MOV AH,AL ; S
          can-Kode
[Y91] 637:          DEC AL
[XYA] 638:          XLAT Byte PTR CS:Table ; Z
          eichen-Kode
[WY6] 639:          CMP AL,OFFH ; k
          ein Kode
[SV2] 640:          JZ FN_FERTG
[AS4] 641: COMMENT $ Akzent
[1MK] 642:          Sollen die drei Akzentzeichen nic
          ht gesondert behandelt
[P2G] 643:          werden, müssen die zwei flgd. Zei
          len gelöscht werden!
[W3M] 644:          Für das Vertauschen der Scan-Kode
          s von z + y entsprechend 3. Zeile
[1R0] 645: END $
[49E] 646:          CALL Akz_Chk ; S
          pezialcheck für Akzent ('')
[AU2] 647:          JC FN_FERTG
[AMF] 648:          CALL YZ_Test ; V
          ertausche Scankodes für Y und Z
[78G] 649: ALT_NORM:CALL PUSH_IT ; s
          chreibe in Tastaturpuffer
[BE8] 650:          JC FERTIG ; B
          ei Überlauf
[GV2] 651: FN_FERTG:CLI
[VS8] 652:          CALL End_Int ; I
          nterrupt freigeben
[86E] 653:          CALL Freigabe ; T
          astatur wieder freigeben
[EN4] 654:          MOV AX,9012H ; ?
[0Y3] 655:          INT 15H ;
[HL2] 656: FERTIG: CLI
[2X9] 657:          POPA ; (
          nur bei AT möglich)
[1D1] 658:          POP ES
[UC1] 659:          POP DS
[CS0] 660:          IRET
[UA8] 661: ;=====
[HDA] 662: ;= Z- und Y-Scan-Kodes vertauschen =
[SAB] 663: ;= ! für die amerikanische Software ! =
[UA8] 664: ;=====
[UV5] 665: YZ_Test: CMP AH,Sc_Z
[P42] 666:          JE Y_PUSH
[8N2] 667:          CMP AH,Sc_Y
[X42] 668:          JE Z_PUSH
[W43] 669: YZ_Ret: RET
[0CE] 670: Y_PUSH: TEST AL,'Z' ; (
          = etwas eingeschränkt)
[F62] 671:          JZ YZ_RET
[8P2] 672:          MOV AH,Sc_Y
[SE0] 673:          RET
[9E8] 674: Z_PUSH: TEST AL,'Y' ; (
          s.O.)
[F62] 675:          JZ YZ_RET
[DP2] 676:          MOV AH,Sc_Z
[SE0] 677:          RET
[UA8] 678: ;=====
[A4B] 679: ;= Akzent-Test und ggf. verarbeiten =
[UA8] 680: ;=====
[LVF] 681: Akz_Chk: CMP AKZENT,false ; I
          F NOT Akzent vorhanden
[YZC] 682:          JE Akz2_Chk ; T
          HEN Check auf Akzent ENDIF
[LOE] 683:          CALL Char_Tst ; I
          F gültiges Folgezeichen
[ZU9] 684:          JNC Akz_Chg ; T
          HEN verarbeiten
[9KB] 685:          PUSH AX ; E
          LSE rette dieses Zeichen

```

```

[ZEC] 686:          MOV AL,[akzent] ;
          hole Akzentzeichen
[SND] 687:          CALL PUSH_IT ;
          schreibe Akzentzeichen
[20D] 688:          POP AX ;
          hole dieses Zeichen zurück
[1ME] 689:          MOV AKZENT,false ;
          setze Akzentflag zurück
[5NB] 690:          JC Akz_fals ; I
          F Pufferüberlauf
[G47] 691:          JMP Akz_Ok ; E
          LSE hier
[V10] 692: ;
[879] 693: Akz2_Chk: CMP AL,27H ; I
          F NOT ''
[ES2] 694:          JZ Akz_Set
[AX1] 695:          CMP AL,' '
[ES2] 696:          JZ Akz_Set
[4X1] 697:          CMP AL,'^'
[0HA] 698:          JNZ Akz_OK ; T
          HEN Normalzeichen
[2VH] 699: Akz_Set: MOV AKZENT,AL ; E
          LSE vermerke Akzentzeichen
[RBM] 700: Akz_fals:STC ; C
          arry:=keine weitere Zeichenverarbeitung
[SE0] 701:          RET
[V10] 702: ;
[RTK] 703: Akz_Chg: MOV BX,offset Akz_Tab1 ; A
          kzentzeichen aus Tabelle
[R66] 704:          XLAT Byte PTR CS:Akz_Tab1
[P3G] 705: Akz_Ok: CLC ; C
          lear:=weitere Verarbeitung
[SE0] 706:          RET
[V10] 707: ;
[RBH] 708: Char_Tst:PUSH BX ; T
          est auf gültiges Folgezeichen
[DH1] 709:          MOV BH,3
[7H1] 710:          MOV BL,0
[AR1] 711:          CMP AL,' '
[8U2] 712:          JZ Char_ok
[BP1] 713:          ADD BL,BH
[PO2] 714:          CMP AL,'ä'
[8U2] 715:          JZ Char_ok
[BP1] 716:          ADD BL,BH
[M12] 717:          CMP AL,'Ä'
[8U2] 718:          JZ Char_ok
[BP1] 719:          ADD BL,BH
[DX1] 720:          CMP AL,'a'
[8U2] 721:          JZ Char_ok
[BP1] 722:          ADD BL,BH
[DU1] 723:          CMP AL,'A'
[8U2] 724:          JZ Char_ok
[BP1] 725:          ADD BL,BH
[LX1] 726:          CMP AL,'c'
[8U2] 727:          JZ Char_ok
[BP1] 728:          ADD BL,BH
[LU1] 729:          CMP AL,'C'
[8U2] 730:          JZ Char_ok
[BP1] 731:          ADD BL,BH
[TX1] 732:          CMP AL,'e'
[8U2] 733:          JZ Char_ok
[BP1] 734:          ADD BL,BH
[TU1] 735:          CMP AL,'E'
[8U2] 736:          JZ Char_ok
[BP1] 737:          ADD BL,BH
[5Y1] 738:          CMP AL,'i'
[8U2] 739:          JZ Char_ok
[BP1] 740:          ADD BL,BH
[MY1] 741:          CMP AL,'n'
[8U2] 742:          JZ Char_ok
[BP1] 743:          ADD BL,BH
[MV1] 744:          CMP AL,'N'
[8U2] 745:          JZ Char_ok
[BP1] 746:          ADD BL,BH
[RY1] 747:          CMP AL,'o'
[8U2] 748:          JZ Char_ok
[BP1] 749:          ADD BL,BH
[9Z1] 750:          CMP AL,'u'
[8U2] 751:          JZ Char_ok
[BP1] 752:          ADD BL,BH
[NZ1] 753:          CMP AL,'y'
[N4B] 754:          JNZ Char_Fal ; s
          onst nicht gültig
[TT4] 755: Char_ok: MOV AL,BL
[TV2] 756:          CMP AKZENT,27H ; e
          rmitteln Tabellenposition
[CB3] 757:          JZ Char_Akz
[NA1] 758:          INC AL
[S73] 759:          CMP AKZENT,' '
[CB3] 760:          JZ Char_Akz
[768] 761:          INC AL ; I

```


NEUE PROFESSIONELLE SHAREWARE FÜR DEN PC



CGA-EMULATOR für die Herculeskarte DM 12
Mit unseren ausgezeichneten CGA-Emulatoren können Sie endlich viele CGA-Grafik erfordernde Programme auf Ihrer Herculeskarte laufen lassen! Warum mehr bezahlen??

SCREEN Utilities DM 12
Viele nützliche Dienstprogramme rund um Ihre Grafikkarte. Einfach Unentbehrlich! Versionen: Hercules Version [] EGA Version []

PC-LOCK - Festplattenschutz DM 12
Perfekter Passwortschutz für jede Festplatte, 100% sichere Softwarelösung, keine Chance für Hacker! Das Booten von Floppy wird verhindert.

FLASHBAK - FESTPLATTENBACKUP DM 12
das professionelle Festplattenbackupprogramm zum sicheren Sichern Ihrer wertvollen Daten und Programme, mit Pull-Down Menüs, automatischem Formatieren der Sicherungsdisketten. Unentbehrlich für alle Festplattenbesitzer. Mit vielen Optionen.

PACKDISK - DISKOPTIMIZER DM12
Der Diskoptimizer bringt Ihre erlarmte Festplatte wieder auf Touren! Sehr empfehlenswert!

STILL RIVER SHELL + AUTOMENU DM 24
sind zwei fantastische Benutzeroberflächen für DOS. DOS Befehle und Programme lassen sich menügesteuert aufrufen. Für ALLE! (2 Disks)

NEWKEY - Tastaturmakros DM 12
Tastaturmakroprozessor. Belegen Sie mit NEWKEY beliebige Tasten mit beliebig langen Texten oder Befehlen, wie Floskeln, Absender, Befehle usw. Sehr empfehlenswert! Ein absolutes Muß vom PC-MAGAZIN in den USA ausgezeichnet! BESTSELLER!

Festplattentools - Nützliche Harddisk Tools DM 12
Die besten Harddisk-Dienstprogramme. Für Festplattenbesitzer mit Durchblick!

PROFESSIONAL MASTER KEY DISK TOOLS DM 12
enthalten viele von den NORTON Utilities Advanced Edition her bekannte Funktionen, wie UNDELETE, UNFORMAT, Diskeditor, Attributes, usw. Warum hunderte DM ausgeben, wenn's auch so geht!

PC-PROMPT - Residente DOS-Hilfe DM 12
Ihre Heilmannchen durch den Dschungel der DOS Befehle. Vollautomatische Hilfe bei der Eingabe von Betriebssystembefehlen. Klasse!

POP-HELP - Residentes Hilfsprogramm DM12
resistente On-Line Hilfe für jedes Programm! Alle Texte können beliebig geändert werden. Bereits inklusive eines DOS 3.x-Manuals. Ein Novum!

DOS*AMATIC - MULTITASKING DM 12
Ermöglicht Multitasking auf dem PC. Zwei Programme laufen gleichzeitig im Hintergrund ab.

HOMEBASE - (Desk-Manager) DM 24
HOMEBASE ist speicherresident, daher können die Funktionen: Datenbank, Texteditor, Rechner, Welter (Termin)kalender, Terminalscheduler, Datenfernübertragung und DOS-Manager aus jedem beliebigen Programm heraus aufgerufen werden.

CHI-WRITER - Wissenschaftliche Texte DM 12
Technisch wissenschaftliche Textverarbeitung mit sechs verschiedenen Schriftarten in WYSIWYG-Darstellung! Der Text wird auf dem Bildschirm genauso angezeigt wie er später ausgedruckt wird! Für CGA-

und unseren CGA-Emulator. Kommerzielle deutsche Version 340-. Bitte Info anfordern!

LO-PRINT - Briefqualität für 9-Nadel Drucker DM 12
Nun können Sie mit Ihrem "alten" 9-Nadeldrucker endlich auch in Briefqualität drucken. Und das auch noch in vielen verschiedenen Schriftarten.

Z-80 CP/M-80 Emulator DM12
Emulation des Z-80 u. CP/M-80 auf dem PC! Viele CP/M-Programme laufen auf dem PC. Klasse!

DESKTEAM - Deskmanager DM 12
Hintergrundprogramm & a Sidekick mit Notizblock, Wecker, Terminkalender, Terminalprogramm, Schreibmaschine, Hilfesystem, DOS-Zugriff.

PC-WRITE 2.4 - TEXTVERARBEITUNG DM 12
Professionelles, superschnelles Textverarbeitungs-system mit: Blocksatz, voller IBM-Zeichensatz, Seitenumschaltung, viele Drucker fertig angepaßt, DOS-Zugriff, Löschen, Verschieben.

GALAXY WORD - TEXTVERARBEITUNG DM12
Extrem schnelles Textsystem der vierten Generation mit Pull-Down Menü. Texte bis RAM-Größe möglich. WordStar kompatibel! Volle Druckersteuerung, auch Laser!

PC-File Plus - Datenbank DM 24
Leicht bedienbare relationale Datenbank mit Textverarbeitung, Maskengenerator, Hilfefunktion, Makros und Passwortschutz. 2 Disks.

WAMPUM - Datenbank im dBase III Format DM 24
Extrem schnelle dBase III kompatible Datenbank. Einfache Menüsteuerung, programmierbare Serienbriefe, Editor, Makros, Maskeneditor. Eines der besten SHAREWARE Programme. 2 Disks

DANCAD3D - 3-DIMENSIONALES CAD DM 24
Exklusiv von uns: Echtes dreidimensionales CAD System mit Animation in Farbe. Das System ermöglicht das Erstellen von 3-D Grafiken mit Animation. Der Kopf glaubt nicht, was die Augen sehen! NEU jetzt für CGA/EGA/Hercules, Laser/24-Nadel-Drucker. **PLATINUM BESTSELLER!** 2 Disks

PC-KEYDRAW - CAD Programm DM 36
Ausgezeichnetes PC CAD System. Die erstellten Grafiken können auf einem Matrixdrucker ausgegeben werden. Eines der besten Grafikprogramme! Für CGA (auch Emulator) und EGA, 3 Disks

ASesAs - Tabellenkalkulation DM 12
Voll Lotus 123* kompatible Tabellenkalkulation mit Grafik, Makros (Recorder), Matrizen, usw. Für CGA, EGA, Hercules, M24. Der Clone des Jahres. Unglaublich, was dieses Programm leistet!

QUBECALC 3-D Tabellenkalkulation DM 12
Mit der Qubecal-Tabellenkalkulation ist es möglich Daten in dreidimensionaler Form anzuordnen und rechnerisch und grafisch auszuwerten. Vom PC Magazin in den USA ausgezeichnet.

PROCOM V2.42 - DFU Programm DM 12
Voll professionelles DFU-Programm mit sämtlichen Raffinessen. DOS-Zugriff, Einsatz als Mailbox möglich, diverse Terminal Emulationen, Programmsprache wie Crossalt, mit allen Protokollen.

WEAK LINK - schnelle PC Kommunikation DM 12
dient zur superschnellen Datenübertragung zwischen zwei IBM-kompatiblen PCs (mit 115200 Baud).

Schnittstelle verbunden und Sie können ganz normale DOS-Befehle zum Kopieren verwenden. Die Laufwerke des einen Rechners können vom anderen PC wie eine RAM-DISK angesprochen werden. Ideal für Computerfirmen. Hervorragend dokumentiert.

3D-CHESS - 3-D Schachprogramm DM 12
Das Profi-Schachprogramm für Anspruchsvolle. Mit voller Grafikdarstellung aller Figuren und Spielzüge. Können Sie es schlagen? Für CGA und Hercules.

SUPER GAMES Grafikspiele für CGA DM 12
Eine Sammlung der beliebtesten Spiele in unserer Bibliothek. Machen Sie sich doch mal ein paar vergnügliche Stunden. Sie haben es verdient!

PC-SPEECH - Das ORIGINAL DM 20
Sprachausgabe für PCs. Ohne Hardwarezusätze kann Ihr PC beliebig lange Texte, in Deutsch, durch den PC Lautsprecher sprechen! Schnittstellen zu BASIC, Turbo Pascal und Assembler.

COMPOSER - Kompositioncomputer DM 12
Mit unserem Composer ist es möglich auf grafischen Notenblättern, Musikstücke zu komponieren, abzuspielen, abzuspeichern, editieren und auch auszudrucken. Für CGA und unseren CGA-Emulator.

PC-TUTOR - DOS Lernprogramm DM 12
PC Neulinge oder Lehrende sollten sich dieses Lehrprogramm für Personalcomputer einmal genau ansehen. Das System führt Sie schrittweise in den Umgang mit dem PC ein.

ARC UTILITIES Prof. Dateikompression DM 12
Die besten Programme zur Dateikompression und Archivverwaltung. Komprimieren Sie Dateien um bis zu 90%.

C-TUTOR - C-LERNPROGRAMM DM 24
Ausgezeichnetes Lernsystem für die Sprache C. Mit vielen Beispielen. (2 Disks) Spezielle TURBO C Version verfügbar. Version bitte angeben!

MODULA2 COMPILER DM 24
Kompletter Modula2 Compiler, incl. Editor, Linker. Mit vielen Beispielen und Programmen und Manual.

MODULA2 TUTOR - Lernprogramm DM 24
Ausgezeichnetes Modula2 Tutor mit vielen Beispielen und Programmen für Modula2.

A86/D86 8086 MACROASSEMBLER DM 24
"The Best Assembler available" so wird A86 in den USA genannt. Ein professioneller Macroassembler für 8086/286. Mit einem hervorragendem Debugger. Ein Muß für alle ernsthaften Assemblerprogrammierer. Teilweise MASM kompatibel.

HP-LASERJET-TOOLS - 5 Disks DM 49
Die HP-Laserjet Tools beinhalten eine Menge neuer Sortförs und ausgesuchter Tools wie residenter Printer Setup für HP Laserjet Drucker und Kompatibel. Ein Muß für alle Laserdruckeranwender! BITTE BEACHTEN SIE, DASS WIR STANDIG NEUE SHAREWAREDISKETTEN FÜR LASERDRUCKEREINSATZ (für MS-WORD, WORDPERFECT, VENTURA PUBLISHER, GEM, (USW.) ANBIETEN! RUFEN SIE UNS AN! HIER FINDEN SIE LASERDISKETTEN ZUM SUPERPREIS (Z.B. 60 PUNKT TIMES ROMAN 12-).

FLODRAW - Fluiddiagramme DM 24
Mit FLOWDRAW können Sie Fluß- oder Organigramme sehr einfach erstellen und auf EPSON FX-80 ausdrucken. (2Disks)

BESONDERS PREISGÜNSTIGE PROGRAMMPAKETE

Die Computer Solutions Programmpakete mit ausgesuchten Disketten zu verschiedenen Themengebieten sind eine ausgezeichnete Gelegenheit, sich mit Super-Software zum Superpreis zu versorgen.

Achtung: Die Pakete bestehen immer aus jeweils 10 Disketten, falls nicht anders angegeben.

ProfiPaket DM 89
zum Kennenlernen. Mit einer Auswahl der besten Programme zur Textverarbeitung, Datenbank, Datenfernübertragung, Tabellenkalkulation, DOS- Hilfsprogrammen und zur Erholung, unseren TOP Spielen. Weiterhin sind enthalten ein Zeichen- und Malprogramm sowie ein ausgezeichnete Computer Tutor.

PC GOLDEN TOOLS DM 89
10er Set mit den besten MS-DOS Utilities zum Retten gelöschter Dateien, Laufwerkstestprogramm: sehr nützlich im Datenschutz: Programme zum Verwalten versteckter Subdirectories; außerdem Super DOS-Nachhilfe mit automatischem Starten von Programmen, residente Kommandozeile (ein Muß!), residenter DOS-Filter, Tastatur Macroprozessor, die BATCH Utilities zeigen Ihnen die verborgenen Fähigkeiten des DOS Batch Kommandos. Nicht nur für Anfänger ist das T-SHELL Menüsystem und das DOS-Online Help gedacht. Die PC GOLDEN TOOLS sollte jeder PC Anwender sein Eigen nennen!

MAGNUM Tools II DM 89
Hier finden Sie: Extended Batch Language zum echten Programmieren in Batch-Dateien, Newkey, BAKERS DOZEN, im Buttons beste Dienstprogramme, PC-PROZENT, DISK COMMANDO, AUTOMENU dem besten DOS-Menüsystem, DISK PRINT SPOOLER mit unbegrenzter Spoolkapazität und PACKDISK. Achten Sie bitte auf die Einzelbeschreibungen. Ein echtes Muß!

MS Tools Deluxe Edition 5 Disks DM 49
Dieses Paket ist absolut unentbehrlich für jeden PC-Nutzer! Mit: Flashbak Plattensicherung, Packdisk Diskoptimierer, Pop-Help Online Hilfe, Shortcut Menu Manager (1Dr. Clone) und Professional Master Key Disk-Tools. Siehe auch obige Beschreibung!

DRUCKEN/TEXT Utilities DM 89
Drucken Sie in Korrespondenzqualität und/oder (17 anderen Schriftarten) auf Ihrem alten Matrixdrucker, mehrspaltig oder in RIESENLETTERN. COLLIST ermöglicht Mehrspaltendruck, Iconmaker erzeugt Grafiken, Sidewriter dient zum Querdruk von breiten Texten oder Tabellen (ähnlich Sideways), raffinierte Dateifilterprogramme runden das Paket ab. Dies und noch sehr viel mehr bietet Ihnen unser Paket DRUCKEN/TEXT-Utilities!

dBase III* Professional Tools DM 89
10 Disketten voller professionell einsetzbarer Sourcecodes bietet Ihnen dieses Paket. Z.B.: Interruptbehandlung aus dBase heraus, C und dBase III usw. Kurz eine Fundgrube für alle dBase III Profis und solche die es werden wollen. Warum dann das Rad ständig neu erfinden? Nutzen Sie diese Tools!

Computergrafik CAD DM 89
Color-Malprogramme, ein komplettes Geschäftsgrafiksystem, Leiterplattendesign, DIA-SHOW, Demo Generator für eigene Demo, Grafik State Of The Art. Für CGA. (teilweise auch CGA-Emulator).

Musik und Komposition DM 89
Verwandeln Sie Ihren PC in einen Kompositionscomputer! Entdecken Sie die Soundfähigkeiten und die Musikalität Ihres Computers! 6 Disks

Deutsche Programme Paket I DM 89
10 Disketten Deutsche Programme: Astrologie, Datenbank mit Editor, Vereinsverwaltung, Tastaturnakros, Datenfernübertragung, Großformatdruck, Buchhaltung, Adressverwaltung und und...

Deutsche Programme Paket II (20 Disks) DM 149
Zweites Paket unserer beliebtesten deutschen Programme, mit dem Inhalt von Paket I und zusätzlich: Dokexpert; RW-FILE; EHO; PC-WINDOW; (resident) ASCII Tabelle, Editor, u.v.m.; Kapitalrechnungen, Clicket, Steuer86, FIX-Faktura das schnelle Programme zu Super-Sparpreis!

Datenfernübertragung DM 89
Die besten DFU-Programme: KERMIT der Klassiker, QMODEM der Vielseitige und PROCOMM der Vollkommit mit allen Übertragungsprotokollen/Terminals-emulationen. RBBS PC ermöglicht es Ihnen, ein eigenes Mailboxsystem aufzubauen! Eine DFU-Utilit-diskette mit äußerst nützlichen Tools rundet dieses außergewöhnliche Paket ab!

LOTUS 1-2-3* Worksheets, Tools DM 89
Power Worksheets! Statistik, Grafik, Datenverwaltung und Finanzen. Tips und Tricks. Werden auch Sie mit unseren vorgefertigten Arbeitsblättern 1-2-3 zum Lotus 1-2-3 Profi!
Wir empfehlen hierzu unseren Lotus 1-2-3 Clone AsCasVs. Alle Lotus 1-2-3 Arbeitsblätter können mit AsCasVs verwendet werden. Klasse!

SYMPHONY Arbeitsblätter und Tools DM 89
Entdecken Sie mit unseren Symphony Tools und Spreadsheets die volle Leistungsfähigkeit dieses Superprogrammes. Die 10 Disketten sind eine Sinfonie in Symphony, die Ihrsgleiches sucht! Themen ähnlich unseren 1-2-3. Paket. Die Symphony Tech Notes sind alleine schon das Geld wert! Dieses Paket sollte wirklich jeder Symphony Anwender sein Eigen nennen. Ideal auch für Firmeneinsatz, da viel Programmierarbeit gespart wird.

Grafikspielesammlung DM 89
Schachprogramm, Arcade Games1, Top Games, Spaceware, Arcade Games2, Willy the Worm, Casino Games, Castle. Für CGA.

TURBO PASCAL ProfiPack DM 89
10 Disks voll Turbo Pascal Sourcecode. Hier sehen Sie wie Könnern programmieren: Interruptbehandlung, Betriebssystemroutinen, File I/O, Grafik, Windowtechnik und residente Programme. Alles meisterhaft programmiert zum Abschauben und Einbauen in eigene Programme. Inclusive Pascal Tutor. Turbo Pascal beherrschen will braucht dieses Paket!

ASSEMBLER Programmierung DM 89
CHASM Assembler, ASMCEM Disassembler, Assembler Tutorial, Assembler/Pascal Tutor, Assembler Tools I + II, MAX Freeware Editor, Wizard Screen

Maskengenerator, Assembler Utilities I + II. Alles im Assembler Quellcode!

C Programmierpaket DM 89
Fabelhaftes Paket mit C Sourcecoderroutinen für fast alle Einsatzgebiete, sogar ein kompletter Editor im C Sourcecode ist darunter! Natürlich verfügt auch dieses Paket über einen ausgezeichneten Tutor dieser Programmiersprache der Zukunft. Mit TURBO PASCAL zu C-Übersetzer! So fällt der Umstieg leicht. Wir HABEN JEDE WOCHNE NEUE C-TOOLS!

Microsoft-QUICKBASIC Tools DM 49
Eine Sammlung nützlicher Routinen und Anwendungen für Ihren Quickbasic Compiler, u.a. einen Fullscreen-Editor! Library Manager usw. 5 Disketten

BASIC Programmierpaket DM 89
Vom strukturierten Programmieren, Optimieren von BASIC Programmen, mathematischen / statistischen Routinen, Cross-Reference Lister, bis hin zum ausgezeichneten BASIC Tutor enthält dieses Paket alles (und noch viel mehr), was das BASIC Programmieren schöner und leichter macht

RBBS-PC - Mailboxprogramm DM 36
Das beste Mailboxprogramm aus den USA. Mit allen Schikanen. RBBS ist ausführlich dokumentiert und wohl der kostengünstigste Weg zu eigenen Mailbox! (3 Disks)

Computer Solutions Software GmbH
Rosenbuschstr. 2 • 8000 München 22
- KEIN LADENVERKAUF -
Telefon: 089/295976 Q
BTX: 089296793

TURBO-PC/AT-BREMSE DM 12
Endlich können Sie auf Ihrem schnellen Turbo PC oder AT auch ältere PC Programme (wichtig z.B. für Spiele) laufen lassen. Der PC-BREMSE zähmt Ihren PC oder AT. Floppy oder Harddisk werden nicht behindert. Ein echtes Muß für alle AT Besitzer!

Neue deutsche Einzelprogramme

EHO - Einzelhandelsorganisation DM 12
Einzelhandelsorganisation per PC. Rechnungen, Umsatzstatistik erledigen. Sie künftig per PC! Daten können mit dBASE oder WAMPUM ausgetauscht werden.

ASTROLOGIEPROGRAMME DM 12
Hervorragendes Astrologieprogramm mit Partnervergleich, Karma, usw....

DOKU-EXPERT-Text-Information DM 12
Relationale Datenbank, Informations-Retrieval-System auf KI-Basis, ähnliches TITAN. DOKU-EXPERT ermöglicht das schnelle Wiederaufrufen von Informationen in großen Datenbeständen. Mit Editor.

ETIKETTENDRUCK DM 12
Dieses Klasse Etikettendruckprogramm macht den Etikettendruck leicht. Gute Druckersteuerung.

PC-BUCH - Finanzbuchhaltung DM 12
Die Buchhaltung für Ihren PC.

VEREINSVERWALTUNG DM 12
Komfortables Verwalten von Vereinsdaten.

CASCADE 2-D CAD System DM 12
Ausgezeichnetes deutsches CAD System für CGA/EGA/Hercules. Ideal für Architekten, Planer und auch Hobbyanwendungen. Maus erforderlich!

ARGUS-Menüprogramm DM 12
DOS-Menüprogramm erleichtert das Verwalten Ihrer Programme auf der Festplatte sehr.

PANORAMAPRINT - Großformatdruck DM 12
Ergaßt Großformat-(Plakat)-Druck von Textdateien.

Bilder und Texte DM 12
Integration von Texten und Blockgrafiken.

RW-FILE Datenbank mit Editor DM 12
Datenbank zum schnellen und komfortablen Verwalten von Daten aller Art. Mit Text- und Maskeneditor z.B. für das Schreiben von Serienbriefen. Sehr empfehlenswert.

Das ist wichtig für Sie

• nur bei **COMPUTER SOLUTIONS** erhalten Sie immer neueste Versionen der SHAREWARE und Public-Domain-Programme. **WIR SIND EUROPAS ERFAHRENSTES SHAREWAREUNTERNEHMEN (SEIT 1982)**

• nur wir bieten Ihnen den **KOSTENLOSEN UPDATE SERVICE** für Disketten, die Sie bei uns gekauft haben. Dazu brauchen Sie uns nur die Originaldiskette im frankierten Rückumschlag einzusenden, dann erhalten Sie gratis die neueste Version ins Haus geschickt.

• gehen Sie auf Nummer sicher und kaufen Sie nur die weißen Disketten mit

Bestellschein

(Kein Ladenverkauf)
Lieferrn Sie uns per/gegen Bar [] Scheck [] Nachnahme: [] Rechnung [] (Bitte ankreuzen)
Die Lieferung gegen Rechnung ist leider nur bei Großfirmen und Institutionen möglich. Ausland nur EUROSCHICK oder BAF.

IBM-PC SHAREWARE Gesamtkatalog (auf 3 Disks) mit Beschreibungen und Stichwortsuchprogramm von ca. 1800 Disketten für nur 12 DM Schutzgebühr [] 3,5"-Format 15 DM []

UNSER KATALOG IST DIE GÜNSTIGSTE MÖGLICHKEIT SICH ÜBER ALLE SHAREWAREPROGRAMME FÜR IHREN IBM-PC ODER KOMPATIBLEN ZU INFORMIEREN. KENNER HABEN IHNI! SIE NICHT? Die Schutzgebühr wird bei Diskettenkauf im Wert von über 100 DM angerechnet (auch nachher).

Senden Sie uns Ihr Gratissimo für [] DOS Extra 3

Rabattstaffel: 1-9 Disks nur DM 12.-/ 10-19 St. DM 10.-/ 20-29 St. DM 8.-/ ab 30 St. auf Anfrage Lieferung auf 3,5 Zoll Disketten 3 DM Aufpreis pro Diskette []

Einzeldisketten: _____

Programmpakete: _____

(Firmenbestellungen bitte mit Firmenpapier ausführen)
Name: _____ Summe: _____ DM
Firma: _____ Versand: _____ 5 DM
Straße: _____ Gesamtsumme: _____ DM
PLZ/Ort: _____ Unterschrift: _____
Telefon: () _____ Datum: _____ 1988


```

[ ] 762: Char_Akz:POP BX
[4SB] 763: MOV AKZENT,false ; s
[ ] 764: CLC ; Z
[UKB] 765: RET ; n
[ ] 766: Char_fal:POP BX
[U37] 767: STC ; F
[SE0] 768: RET
[UA8] 769: ;=====
[F9A] 770: ;= Test auf CTRL-ALT-DEL, ggf. booten =
[UA8] 771: ;=====
[YLC] 772: BOOT: TEST BYTE PTR [KB_FLAG],(ALT_SHIF
T + CTL_SHIFT)
[SW9] 773: JZ C_BL1
[ ] 774: ; weder ALT noch CTL
[BP7] 775: TEST BYTE PTR [KB_FLAG],CTL_SHIFT
[TR8] 776: JZ No_Boot
[ ] 777: ; alt + del
[FP7] 778: TEST BYTE PTR [KB_FLAG],ALT_SHIFT
[668] 779: JNZ Alt_Int
[ ] 780: No_Boot: JMP FN_FERTG
[1FK] 781: ; alternativ: Kein Boot möglich!
[V10] 782: Alt_Int: JMP Ext_9H
[V6C] 783: ; Boot über BIOS
[UA8] 784: ;=====
[BP9] 785: ;= Sondertaste gedrückt: =
[NE7] 786: ;= - ohne ALT - ohne Ctrl =
[BY9] 787: ;= - mit Links- oder Rechtsschift =
[UA8] 788: ;=====
[T03] 789: RT_LR_SHIFT:
[CT7] 790: TEST BYTE PTR [KB_FLAG],NUM_STATE
[Z22] 791: JZ RT_LR_2
[833] 792: JMP MOV1_EXT
[V10] 793: ;
[PK5] 794: RT_LR_2: JMP MOV2_EXT
[UA8] 795: ;=====
[BSB] 796: ;= Normale Taste verarbeiten unter =
[7NB] 797: ;= Berücksichtigung von KB_Flag =
[UA8] 798: ;=====
[EDF] 799: NG_TABL: CMP AL,Sc_F1 ; Z
[ ] 800: uerst Sondertasten-Test
[349] 801: JB NORMAL ;
[ ] 802: ggf. umlenken
[Z53] 803: CMP AL,Sc_F10
[TV2] 804: JBE FUNC_TAB
[663] 805: CMP AL,Sc_F11
[GS2] 806: JE FUNC_TAB
[D63] 807: CMP AL,Sc_F12
[GS2] 808: JE FUNC_TAB
[305] 809: CMP AL,Sc_Gr_minus
[UW2] 810: JZ GRY_SIGN
[2K4] 811: CMP AL,Sc_Gr_plus
[ZN2] 812: jz normal
[ZW4] 813: JZ Alt_Int
[HW3] 814: CMP AL,Sc_Del
[3G1] 815: JZ BOOT
[XNG] 816: CMP AL,Sc_Sys_Req ; (
[ ] 817: evtl. damit neue Tabelle öffnen)
[Z12] 818: JZ NORMAL
[1T9] 819: C_BL1: TEST BYTE PTR [KB_FLAG],ALT_SHIFT
[3M1] 820: JZ C_BL2
[782] 821: JMP RT_ALT
[4T9] 822: C_BL2: TEST BYTE PTR [KB_FLAG],CTL_SHIFT
[C43] 823: JNZ MOV3_EXT
[8AB] 824: TEST BYTE PTR [KB_FLAG],(LEFT_SHI
FT + RIGHT_SHIFT)
[XS3] 825: JNZ RT_LR_SHIFT
[CT7] 826: TEST BYTE PTR [KB_FLAG],NUM_STATE
[N33] 827: JNZ MOV2_EXT
[ON7] 828: MOV1_EXT:MOV BX,offset TABLE
[YK2] 829: JMP COD_EXT
[V10] 830: ;
[BEA] 831: GRY_SIGN:TEST BYTE PTR [KB_FLAG],ALT_SHIFT
[Z12] 832: JZ NORMAL
[BP7] 833: TEST BYTE PTR [KB_FLAG],CTL_SHIFT
[Z12] 834: JZ NORMAL
[XG2] 835: JMP CURSIZE
[V10] 836: ;
[LCA] 837: FUNC_TAB:TEST BYTE PTR [KB_FLAG],ALT_SHIFT
[P63] 838: JNZ MOVX_EXT
[BP7] 839: TEST BYTE PTR [KB_FLAG],CTL_SHIFT
[C43] 840: JNZ MOV3_EXT
[8AB] 841: TEST BYTE PTR [KB_FLAG],(LEFT_SHI
FT + RIGHT_SHIFT)
[KZ2] 842: JZ MOV1_EXT

```

```

[KH7] 837: MOV2_EXT:MOV BX,offset TABL2
[YK2] 838: JMP COD_EXT
[V10] 839: ;
[CY9] 840: MOV3_EXT:MOV BX,offset TABL3 ; C
[ ] 841: JMP COD_EXT
[YK2] 842: ;
[KRA] 843: MOVX_EXT:MOV BX,offset TABL_X ; A
[ ] 844: JMP COD_EXT
[V10] 845: ;
[UZC] 846: NORMAL: CMP AL,Sc_Prt_Scr ; N
[ ] 847: JNZ NORM_CON
[A33] 848: JMP Chk_Prt
[EX2] 849: ;
[ZDA] 850: NORM_CON:TEST BYTE PTR [KB_FLAG],ALT_SHIFT
[T33] 851: JNZ MOV4_NOR
[BP7] 852: TEST BYTE PTR [KB_FLAG],CTL_SHIFT
[R33] 853: JNZ MOV3_NOR
[8AB] 854: TEST BYTE PTR [KB_FLAG],(LEFT_SHI
FT + RIGHT_SHIFT)
[7N2] 855: JNZ INV_NOR
[5Y7] 856: TEST BYTE PTR [KB_FLAG],CAPS_STAT
E
[XY2] 857: JZ MOV1_NOR
[ON2] 858: CALL Letter
[HX2] 859: JC MOV2_NOR
[RM7] 860: MOV1_NOR:MOV BX,offset TABLE
[T03] 861: JMP COD_NORM
[V10] 862: ;
[9H7] 863: MOV2_NOR:MOV BX,offset TABL2
[T03] 864: JMP COD_NORM
[V10] 865: ;
[LH7] 866: MOV3_NOR:MOV BX,offset TABL3
[T03] 867: JMP COD_NORM
[UA8] 868: ;=====
[9Y7] 869: ;= Alt-Verarbeitung =
[5S8] 870: ;= Extend Kode für: A-Z, a-z, 0-9 =
[GS4] 871: ;= F1-F12 =
[698] 872: ;= Normal Kode für: sonst =
[UA8] 873: ;=====
[6DA] 874: MOV4_NOR:TEST BYTE PTR [KB_FLAG],CTL_SHIFT
[V33] 875: JNZ MOV5_NOR
[8AB] 876: TEST BYTE PTR [KB_FLAG],(LEFT_SHI
FT + RIGHT_SHIFT)
[P13] 877: JNZ MOV5NOR
[UH1] 878: PUSH AX
[BU5] 879: CMP AL,Sc_F1 ; <
[ ] 880: F1
[CC2] 881: JB ALT_CNT
[2B6] 882: CMP AL,Sc_F10 ; <
[ ] 883: = F10
[RL2] 884: JBE MOV_ATX
[V86] 885: CMP AL,Sc_F11 ; =
[ ] 886: F11
[EG2] 887: JE MOV_ATX
[996] 888: CMP AL,Sc_F12 ; =
[ ] 889: F12
[EG2] 890: JE MOV_ATX
[MN7] 891: ALT_CNT: MOV BX,offset TABLE
[6T1] 892: MOV AH,AL
[Y91] 893: DEC AL
[OSC] 894: XLAT Byte PTR CS:Table ; d
[ ] 895: ies ist einfacher,
[8E8] 896: CMP AL,'0' ; d
[ ] 897: a Scan-Kodes
[7ZB] 898: JB MOV_ALT ; a
[ ] 899: nders sortiert sind
[NT1] 900: CMP AL,'9'
[RL2] 901: JBE MOV_ATX
[DM7] 902: AND AL,11011111B ; (
[ ] 903: a-z) -> (A-Z)
[DU1] 904: CMP AL,'A'
[ME2] 905: JB MOV_ALT
[SW1] 906: CMP AL,'Z'
[KE2] 907: JA MOV_ALT
[LM3] 908: MOV_ATX: POP AX
[H66] 909: MOV BX,offset TABL_A4
[YK2] 910: JMP COD_EXT
[LB4] 911: COMMENT $ MOV_ALT:
[FZE] 912: Da hier nicht mehr viel übrig ble
ibt, könnte man
[KPE] 913: die Tabelle kleiner machen, aber
auf Grunde des
[UDF] 914: Aufbaues des Scan-Kodes, würde di
e Abfrage kaum
[Z6G] 915: weniger Speicherplatz verbrauchen
, als die Tabelle.
[70G] 916: Der Aufwand stände in keinem Verh

```

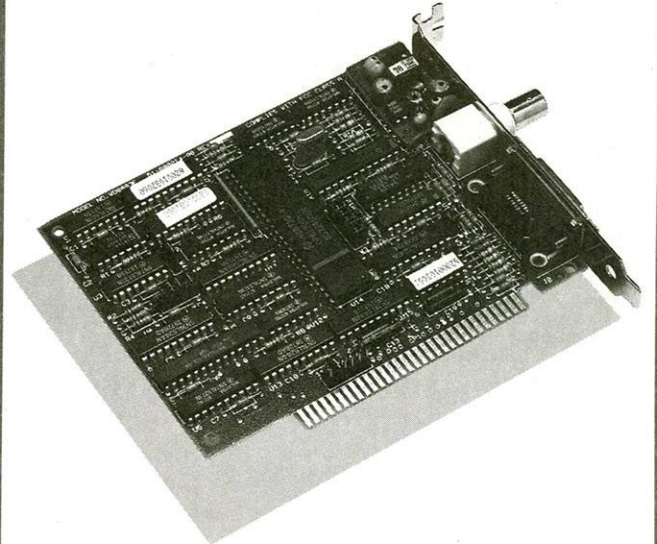


```

    ältnis zum Gewinn.
[1R0] 909: END $
[7K3] 910: MOV_ALT: POP    AX
[885] 911:      MOV    BX,offset TABL4
[T03] 912:      JMP    COD_NORM
[V10] 913: ;
[998] 914: MOVS4NOR:MOV    BX,offset TABL_S4
[T03] 915:      JMP    COD_NORM
[E4E] 916: MOV5_NOR:TEST   BYTE PTR [KB_FLAG],(LEFT_SHI
    FT + RIGHT_SHIFT)
[3X2] 917:      JZ     MOV5NOR
[D85] 918:      MOV    BX,offset TABL5
[T03] 919:      JMP    COD_NORM
[V10] 920: ;
[E98] 921: MOVS5NOR:MOV    BX,offset TABL_S5
[T03] 922:      JMP    COD_NORM
[V10] 923: ;
[1RA] 924: INV_NOR: TEST   BYTE PTR [KB_FLAG],CAPS_STAT
    E
[ZYZ] 925:      JZ     MOV2_NOR
[ON2] 926:      CALL   Letter
[FX2] 927:      JC     MOV1_NOR
[N23] 928:      JMP    MOV2_NOR
[UA8] 929: ;=====
[YZB] 930: ;= Print-Screen-Tastenbehandlung =
[UA8] 931: ;=====
[LSE] 932: Chk_Prt: TEST   BYTE PTR [KB_FLAG],(LEFT_SHI
    FT + RIGHT_SHIFT)
[783] 933:      JNZ    Prt_Con1
[BP7] 934:      TEST   BYTE PTR [KB_FLAG],CTL_SHIFT
[D83] 935:      JNZ    Prt_Con2
[8T1] 936:      MOV    AL,'*'
[2R4] 937:      MOV    AH,Sc_Prt_Scr
[V03] 938:      JMP    ALT_NORM
[TH0] 939: ;
[80B] 940: Prt_Con1:TEST   BYTE PTR [KB_FLAG],CTL_SHIFT
[04A] 941:      JNZ    HARDCOPY          ; S
    hift-Ctl-PrtScr
[RT1] 942:      MOV    AL,'/'
[2R4] 943:      MOV    AH,Sc_Prt_Scr
[V03] 944:      JMP    ALT_NORM
[V10] 945: ;
[8EC] 946: Prt_Con2:MOV    AL,Sc_Extended          ; C
    tl-PrtScr
[4Z1] 947:      MOV    AH,72H
[V03] 948:      JMP    ALT_NORM
[UA8] 949: ;=====
[WKN] 950: ;= Druckerausgabe des Bildschirmtextes wege
    n SHIFT-PrtSc im TEXT-Modus =
[1ED] 951: ;= Ruft den Interrupt 05H auf
    =
[B5P] 952: ;= (Hier könnte jetzt ein eigener EGA-Druck
    ertreiber, etc. eingebunden =
[6ZM] 953: ;= werden; dann kein INT5-Aufruf; HardCopyF
    lag testen und setzen) =
[UA8] 954: ;=====
[PNG] 955: HARDCOPY:CALL   Freigabe          ; T
    astatur wieder freigeben
[VSB] 956:      CALL   End_Int          ; I
    nterrupt freigeben
[VN9] 957:      PUSH   BP          ; B
    P-Register retten
[4FB] 958:      INT     05          ; S
    creen-Dump ausführen
[URA] 959:      POP     BP          ; B
    P wieder herstellen
[F42] 960:      JMP     FERTIG
[UA8] 961: ;=====
[DE8] 962: ;= CTRL-NumLock = PAUSE =
[UA8] 963: ;=====
[0EA] 964: PAUSE: OR     BYTE PTR [KB_2_FLAG],HOLD_ST
    ATE
[86E] 965:      CALL   Freigabe          ; T
    astatur wieder freigeben
[VSB] 966:      CALL   End_Int          ; I
    nterrupt freigeben
[066] 967:      CMP     BYTE PTR [CRT_Mode],07
[6K2] 968:      JZ     PAUS_LP
[7Z5] 969:      MOV     DX,CS:Video_Port
[LA6] 970:      MOV     AL,[CRT_Mode_Set]
[3U1] 971:      OUT     DX,AL
[HDB] 972: PAUS_LP: TEST   BYTE PTR [KB_2_FLAG],HOLD_ST
    ATE
[WP2] 973:      JNZ     PAUS_LP
[F42] 974:      JMP     FERTIG
[UA8] 975: ;=====
[5A8] 976: ;= CTRL-ScrLock = BREAK =
[NR9] 977: ;= löscht den Puffer und führt =
[8D8] 978: ;= den Interrupt 1BH aus =
[UA8] 979: ;=====

```

Kleine Karte für großartige Netzwerktechnik



Wie bringt ein IBM PC, XT, AT oder PS/2 auch im Netzverbund seine volle Leistung? Mit EtherCard PLUS von Western Digital, dem Ethernet Adapter-Board mit den einzigartigen Leistungsmerkmalen:

- Dual-Port 8 K-Memory für höchsten Durchsatz
- 10-Mbps-Betrieb, CSMA/CD
- Auch kompatibel mit IEEE 802.3
- Transceiver für »Thin Ethernet« auf der Karte
- Keine DMA-Kanäle erforderlich
- High Performance, low Power CMOS-LAN-Chip
- Software für ViaNet, Novell und NetBIOS/OSI
- TCP/IP-Protokoll als Option

Fordern Sie unsere ausführlichen
Unterlagen an!

WESTERN DIGITAL
DEUTSCHLAND GMBH

Zamdorfer Straße 26 · 8000 München 80
Tel. (089) 9101071 · Telex 5214568 · Fax (089) 914611


```

[D4E] 980: BREAK: MOV AX,[Buffer_Start] ; C
          lea buffer
[EV5] 981: MOV [Buffer_Tail],AX
[2S5] 982: MOV [Buffer_Head],AX
[FWF] 983: MOV BYTE PTR [BiosBreakFlag],MSB
          ; Ctrl-Break-Flag im BIOS
[86E] 984: CALL Freigabe ; T
          astatur wieder freigeben
[1Y2] 985: INT Abbruch
[9U1] 986: XOR AX,AX
[V03] 987: JMP ALT_NORM
[UA8] 988: ;=====
[20C] 989: ;= ALT-Cursor-Block-Zahlenverarbeitung =
[33C] 990: ;= (Byte-Operation ohne Überlauf-test) =
[UA8] 991: ;=====
[GM3] 992: RT_ALT: PUSH AX
[Y91] 993: DEC AL
[Y75] 994: MOV BX,offset TABL2
[VZ5] 995: XLAT Byte PTR CS:Table
[GS1] 996: SUB AL,'0'
[AVA] 997: JC No_num ; I
          F ASCII(WERT) < ASCII('0')
[KG1] 998: CMP AL,9
[9K9] 999: JA No_num ; I
          F ASCII(WERT-'0') > 9
[CC1] 1000: POP BX
[V59] 1001: MOV BL,AL ; B
          L:=neuer Wert
[MAF] 1002: MOV AL,[Alt_Input] ; A
          L:=bisheriger Speicherwert
[9PG] 1003: IMUL AX,10 ; (
          Integermultiplik. nur bei AT möglich)
[X87] 1004: ADD AL,BL ; A
          L:=(AL*10)+BL
[VNE] 1005: MOV [Alt_Input],AL ; n
          euer Wert in Bytespeicher
[EZ2] 1006: JMP FN_FERTG
[V10] 1007: ;
[B8A] 1008: No_num: POP AX ; k
          eine Zahl
[T23] 1009: JMP MOV4_NOR
[UA8] 1010: ;=====
[BC7] 1011: ;= Erst HOLD-Test =
[S27] 1012: ;= TRUE:= NUMLOCK-Test =
[BU9] 1013: ;= wenn nicht, HOLD aufheben =
[S6B] 1014: ;= FALSE:= Beginn Tastenverarbeitung =
[UA8] 1015: ;=====
[5FD] 1016: NG_TASTE:TEST BYTE PTR [KB_2_FLAG],HOLD_ST
          ATE; IF PAUSE
[KLE] 1017: JNZ RESET_HLD
          ; THEN then PAUSE-Verarbeitung
[RXD] 1018: JMP NG_TABL
          ; ELSE zum Sprungverteiler
[V10] 1019: ;
[6VD] 1020: RESET_HLD:
          ; System im Pausemodus
[6EC] 1021: CMP AL,Sc_Num_lock
          ; IF wieder NUM-LOCK
[VPB] 1022: JE HLD_CON
          ; THEN weiter mit PAUSE
[DCC] 1023: AND BYTE PTR [KB_2_FLAG],0F7H
          ; ELSE NOT HOLD STATE
[D5G] 1024: HLD_CON: JMP FN_FERTG
          ; und kein Zeichen in Puffer
[UA8] 1025: ;=====
[2FB] 1026: ;= FUNC TEST testet auf Sondertasten, =
[02B] 1027: ;= für diese findet eine erste Über- =
[089] 1028: ;= setzung statt: AX bleibt erhalten =
[089] 1029: ;= Einsprung: AL = Scan-Kode =
[33B] 1030: ;= Rückgabe: Carry: gesetzt = FN-Taste=
[EW5] 1031: ;= BL:=XLAT(AL) =
[DC8] 1032: ;= sonst Norm-Taste =
[UA8] 1033: ;=====
[294] 1034: FUNC_TEST PROC NEAR
[68F] 1035: AND AL,NOT MSB ; m
          askiere MSB aus wegen Tabelle
[4Z3] 1036: CMP AL,Sc_Alt
[NY2] 1037: JZ FUNC_CRY
[6Z3] 1038: CMP AL,Sc_Ctl
[NY2] 1039: JZ FUNC_CRY
[RB4] 1040: CMP AL,Sc_L_Shift
[NY2] 1041: JZ FUNC_CRY
[NC4] 1042: CMP AL,Sc_R_Shift
[NY2] 1043: JZ FUNC_CRY
[H04] 1044: CMP AL,Sc_Ins
[F22] 1045: JZ FN_INS
[925] 1046: CMP AL,Sc_Num_lock
[NY2] 1047: JZ FUNC_CRY
[Y94] 1048: CMP AL,Sc_Scroll
[NY2] 1049: JZ FUNC_CRY
[GW3] 1050: CMP AL,Sc_Caps
[NY2] 1051: JZ FUNC_CRY
[4P2] 1052: FN_RET: CLC

[SE0] 1053: RET
[V10] 1054: ;
[ZM7] 1055: FUNC_CRY:MOV BX,offset TABLE
[UH1] 1056: PUSH AX
[Y91] 1057: DEC AL
[VZ5] 1058: XLAT Byte PTR CS:Table
[DT1] 1059: MOV BL,AL
[5C1] 1060: POP AX
[1X2] 1061: FNCY_RET:STC
[SE0] 1062: RET
[V10] 1063: ;
[T9A] 1064: FN_INS: TEST BYTE PTR [KB_FLAG],NUM_STATE
[072] 1065: JNZ FN_RET
[6W3] 1066: MOV BL,INS_STATE
[923] 1067: JMP FNCY_RET
[BL3] 1068: FUNC_TEST ENDP
[1L5] 1069: ;=====
[5TB] 1070: ;= Tastatur-Kommandoport-Behandlung -
[1L5] 1071: ;=====
[WD3] 1072: Freigabe:CLI
[6SA] 1073: MOV AH,10101110B ; T
          astatur freigeben
[T33] 1074: JMP PRT_CONT
[V10] 1075: ;
[KL3] 1076: Sperren: CLI
[1DA] 1077: MOV AH,10101101B ; T
          astatur sperren
[TS6] 1078: PRT_CONT:IN AL,kb_Command
[G42] 1079: TEST AL,02H
[T43] 1080: JNZ PRT_CONT
[US1] 1081: MOV AL,AH
[NAF] 1082: OUT kb_Command,AL ; T
          astaturkommando ausgeben
[PE0] 1083: STI
[SE0] 1084: RET
[1L5] 1085: ;=====
[PZA] 1086: ;=Rückgabe: Carry wenn Kleinbuchst. -
[1L5] 1087: ;=====
[5B4] 1088: Letter PROC NEAR
[UH1] 1089: PUSH AX
[XA5] 1090: MOV BX,offset TABLE
[Y91] 1091: DEC AL
[VZ5] 1092: XLAT Byte PTR CS:Table
[P02] 1093: CMP AL,'ä'
[252] 1094: JE Lettr2
[622] 1095: CMP AL,'ö'
[252] 1096: JE Lettr2
[D02] 1097: CMP AL,'ü'
[252] 1098: JE Lettr2
[DX1] 1099: CMP AL,'a'
[W42] 1100: JB Lettr2
[NF2] 1101: CMP AL,'z'+1
[6D0] 1102: CMC
[RY2] 1103: Lettr2: CMC
[5C1] 1104: POP AX
[SE0] 1105: RET
[VF3] 1106: Letter ENDP
[UA8] 1107: ;=====
[X18] 1108: ;= Cursor setzen =
[UA8] 1109: ;=====
[T14] 1110: CURSIZE PROC NEAR
[UH1] 1111: PUSH AX
[8K1] 1112: PUSH CX
[1K1] 1113: PUSH BP
[VU1] 1114: XOR CX,CX
[021] 1115: MOV CH,10H
[8AB] 1116: TEST BYTE PTR [KB_FLAG],(LEFT_SHI
          FT + RIGHT_SHIFT)
[WXA] 1117: JNZ Cur_Jmp ; C
          ursor ein/Größe
[HV7] 1118: XOR AX,AX ; C
          ursor aus
[KV5] 1119: MOV CX,[CURSOR_MODUS]
[EK8] 1120: CMP CX,WORD PTR [Cursor_Mode]
[MT1] 1121: MOV CH,AL
[DX8] 1122: JZ Cur_Jmp ; C
          ursor full
[ST1] 1123: MOV CH,CL
[CE8] 1124: DEC CH ; C
          ursor small
[DM6] 1125: Cur_Jmp: MOV AH,CurGr
[P42] 1126: INT Video
[CC1] 1127: POP BP
[LC1] 1128: POP CX
[5C1] 1129: POP AX
[EZ2] 1130: JMP FN_FERTG
[F63] 1131: CURSIZE ENDP
[B86] 1132: SUBTTL Installation
[X21] 1133: PAGE +
[599] 1134: ;=====
[1ZD] 1135: ;= Installiere den Tastaturtreiber

```



```

[599] 1136: ;-----
[B05] 1137: INITIALIZE PROC NEAR
[W80] 1138: ;--
[S27] 1139: ;-- Speichere alte Vektoren
[W80] 1140: ;--
[8K1] 1141: PUSH CS
[UMD] 1142: POP DS ; DS:=auf di
      eses Programmsegment!
[YNB] 1143: MOV AH,GetVec ; hole Inter
      ruptvektor
[824] 1144: MOV AL,Tast_Int
[9WA] 1145: INT MsDos ; in ES:BX
      und speichern
[C46] 1146: MOV Old9H_Vector[0],bx
[956] 1147: MOV Old9H_Vector[2],es
[YNB] 1148: MOV AH,GetVec ; hole Inter
      ruptvektor
[X23] 1149: MOV AL,Time_Int
[9WA] 1150: INT MsDos ; in ES:BX
      und speichern
[516] 1151: MOV Old1CH_Vector[0],bx
[A16] 1152: MOV Old1CH_Vector[2],es
[W80] 1153: ;--
[1F9] 1154: ;-- Teste ob schon installiert
[W80] 1155: ;--
[H56] 1156: MOV ES,Old9H_Vector[2]
[BD6] 1157: MOV DI,offset KENNUNG
[1X1] 1158: MOV SI,DI
[2N5] 1159: MOV CX,(VERSION-KENNUNG)
[7D0] 1160: CLD
[668] 1161: REPE CMPSB ; ES:DI <->
      DS:SI
[MW2] 1162: JNE Do_Init
[A2B] 1163: JMP No_Init ; bereits in
      stalliert
[W80] 1164: ;--
[ZHG] 1165: ;-- Teste ob ScreenSaver gewünscht, ggf. Fla
      gs setzen
[W80] 1166: ;--
[7G9] 1167: Do_Init: MOV Screen_Save,false
[S04] 1168: MOV BX,CmdLine
[T75] 1169: MOV AL,Byte PTR [BX]
[NR9] 1170: AND AX,00FFH ; AH:=0 und
      Test auf AL>0
[X23] 1171: JNZ ParTest
[XG6] 1172: ParErr: JMP Wron_Par
[V10] 1173: ;
[Y3A] 1174: ParTest: ADD BL,AL ; ([al]+[bl]
      < 0FFH)
[T75] 1175: MOV AL,Byte PTR [BX]
[GS1] 1176: SUB AL,'0'
[ZR9] 1177: JC ParErr ; Fehler bei
      : ASCII < 48='0'
[HHE] 1178: JZ NoSave ; bei '0' ke
      ine Installation gewünscht
[T6B] 1179: CMP AL,9 ; 30 Sek. bi
      s zu 1 Std. möglich
[PA2] 1180: JA ParErr
[TS1] 1181: XOR AH,AH
[NA1] 1182: INC AX
[H96] 1183: MOV Screen_Save,true
[16F] 1184: IMUL AX,(546*Sekund) ; (18,2*30 s
      ec = 546 ist 1/2 Minute)*Sekund
[SCK] 1185: MOV Screen_Tick,AX ; Festwert w
      ird auf Screen_Count kopiert
[ZTG] 1186: MOV Screen_Count,AX ; Inhalt wir
      d auf 0 runtergezählt
[PB4] 1187: NoSave LABEL NEAR
[W80] 1188: ;--
[S4K] 1189: ;-- Variableninitialisierung (a. Video-Contr
      ollers-Adresse)
[W80] 1190: ;--
[CU3] 1191: MOV adapter,3
[6R3] 1192: MOV AX,BIOS_DATA
[6V1] 1193: MOV ES,AX
[Z76] 1194: MOV DI,offset ADDR_6845
[YC3] 1195: MOV DX,ES:[DI]
[7E9] 1196: MOV CRTC_ADR,DX ; 6845 Index
      Register
[5F1] 1197: ADD DX,4
[ZSE] 1198: MOV Video_Port,DX ; 6845 Displ
      ay Mode Control Port
[W80] 1199: ;--
[3F4] 1200: ;-- Test auf EGA
[W80] 1201: ;--
[BA6] 1202: MOV cursor_modus,0C0DH
[GV8] 1203: MOV AX,0C00H ; ES:=EGA BI
      OS Segment
[6V1] 1204: MOV ES,AX
[X5B] 1205: MOV DI,1EH ; DI zeigt a
      uf EGA-Signatur

```

```

[PSC] 1206: MOV SI,offset IBM ; SI zeigt a
      uf 'IBM'-Text
[FZA] 1207: MOV CX,3 ; CX:=Anzahl
      der Buchstaben
[7D0] 1208: CLD
[W80] 1209: REPE CMPSB ; WENN ident
      isch,
[UWA] 1210: JE INIT1 ; DANN gehe
      zu EGA-Behandlung
[VK9] 1211: DEC adapter ; SONST hier
      weiter
[W80] 1212: ;--
[136] 1213: ;-- Test ob MDA oder CGA
[W80] 1214: ;--
[S56] 1215: MOV cursor_mode,0607H
[P6A] 1216: MOV AH,ScrGet ; hole Video
      -Modus
[P42] 1217: INT Video
[UZ8] 1218: CMP AL,7 ; WENN NICHT
      monochrom
[FOB] 1219: JNE INIT1 ; DANN gehe
      zu CGA-Behandlung
[VK9] 1220: DEC adapter ; SONST hier
      weiter
[W80] 1221: ;--
[1PN] 1222: ;-- Test auf Hercules nicht implementiert (a
      ngenommen identisch mono)
[W80] 1223: ;--
[943] 1224: DEC adapter
[W80] 1225: ;--
[557] 1226: ;-- Monochrom + Hercules
[W80] 1227: ;--
[BA6] 1228: MOV cursor_modus,0C0DH
[765] 1229: SUB VIDEO_SEGMENT,0800H
[W80] 1230: ;--
[X27] 1231: ;-- setze Cursorgröße
[W80] 1232: ;--
[L18] 1233: INIT1: MOV CX,cursor_modus
[W80] 1234: MOV AH,CurGr
[P42] 1235: INT Video
[W80] 1236: ;--
[W88] 1237: ;-- Setze die neuen Vektoren
[W80] 1238: ;--
[8K1] 1239: PUSH CS
[UC1] 1240: POP DS
[U4C] 1241: MOV AH,SetVec ; setze Inte
      ruptvektor
[824] 1242: MOV AL,Tast_Int
[418] 1243: MOV DX,offset NEW9H ; aus DS:DX
[R22] 1244: INT MsDos
[LS6] 1245: OR Screen_Save,false
[U8F] 1246: JE No_Vec1C ; Screensave
      r war nicht gewünscht
[U4C] 1247: MOV AH,SetVec ; setze Inte
      ruptvektor
[X23] 1248: MOV AL,Time_Int
[6R5] 1249: MOV DX,offset NEW1CH
[R22] 1250: INT MsDos
[4C4] 1251: No_Vec1C LABEL NEAR
[W80] 1252: ;--
[BGE] 1253: ;-- Terminiere Programm, ggf. mit Fehlermeld
      ung
[W80] 1254: ;--
[6V7] 1255: MOV DX,offset initialize+1
[0BE] 1256: INT Terminate ; ENDE: Prog
      ramm bleibt resident
[LK4] 1257: ;
[FS9] 1258: Wron_Par: mov dx,offset ErrMsg1 ; ODER
[UA4] 1259: mov ah,StrgOut
[R82] 1260: int MsDos
[8W9] 1261: No_Init: mov dx,offset ErrMsg2
[UH4] 1262: mov ah,StrgOut
[R82] 1263: int MsDos
[9N8] 1264: mov dl,Bell ; Fehlersign
      al
[9F4] 1265: mov ah,CharOut
[R82] 1266: int MsDos
[X99] 1267: INT Pend ; ENDE: nich
      t resident
[V10] 1268: ;
[85L] 1269: ErrMsg1 DB 'Parameterangabe im Tastatur
      treiber falsch.',CR,LF
[F19] 1270: DB 'Aufruf: 'T3 n' n {0..9}',CR
      ,LF,'$'
[A5D] 1271: ErrMsg2 DB 'Nicht (erneut) installiert!
      ',CR,LF,'$'
[2G3] 1272: INITIALIZE ENDP
[V10] 1273: ;
[1M2] 1274: CSEG ENDS
[PC2] 1275: END Beginn

```

Listing 1. Assembler-Quellcode des DOS-T3-Tastaturtreibers


```

[B87] 0: E9 82 09 90 00 00 00 00 00 00 00 00 49 42 4D 00
[2F7] 1: 00 00 00 00 00 B8 28 63 29 20 32 34 2E 30 32 2E
[BP7] 2: 31 39 38 38 20 4B 48 53 20 4B 94 6C 6E 2C 20 54
[4C7] 3: 65 6C 2E 20 30 32 32 31 2D 33 31 37 32 33 32 20
[W27] 4: 56 33 2E 31 31 24 00 00 00 00 00 00 00 00 00 00
[9G7] 5: 1B 31 32 33 34 35 36 37 38 39 30 E1 27 08 09 71
[3S7] 6: 77 65 72 74 7A 75 69 6F 70 81 2B 0D 04 61 73 64
[9R7] 7: 66 67 68 6A 6B 6C 94 84 3C 02 23 79 78 63 76 62
[XV7] 8: 6E 6D 2C 2E 2D 01 2A 08 20 40 3B 3C 3D 3E 3F 40
[YR7] 9: 41 42 43 44 20 10 47 48 49 2D 4B FF 4D 2B 4F 50
[4U7] 10: 51 52 53 2A FF FF 57 58 1B 21 22 15 24 25 26 2F
[AP7] 11: 28 29 3D 3F 60 08 00 51 57 45 52 54 5A 55 49 4F
[ZU7] 12: 50 9A 2A 0D 04 41 53 44 46 47 48 4A 4B 4C 99 8E
[UW7] 13: 3E FF 5E 59 58 43 56 42 4E 4D 3B 3A 5F 01 00 08
[928] 14: 20 FF 54 55 56 57 58 59 5A 5B 5C 5D FF FF 37 38
[2T7] 15: 39 2D 34 35 36 2C 31 32 33 30 2E 2F FF FF 37 38
[OP8] 16: 0B 00 40 FF FF FF 1E FF FF FF FF FF 1F FF 7F 11
[4V7] 17: 17 05 12 14 1A 15 09 0F 10 7B 7D 0A 0F 01 13 04
[XL7] 18: 06 07 08 0A 0B 0C 40 40 7C FF 1C 19 18 03 16 02
[6G8] 19: 0E 0D FF FF FF FF FF FF 20 FF 5E 5F 60 61 62 64
[GC8] 20: 64 65 66 67 FF FF 77 FF 84 2D 73 FF 74 2B 75 FF
[C99] 21: 76 FF FF 2F FF FF FF FF 1B FF FF FF FF FF FF FF
[969] 22: FF FF FF A8 AE 08 09 FF FF FF FF FF FF FF 7B 7D
[OT8] 23: FF 5B 5D 0D 04 FF FF FF FF FF FF FF FF 7B 7D
[FE8] 24: 5C 02 23 FF FF FF FF FF FF 2C 2E 2D 01 2A 08
[CL8] 25: 20 40 FF FF FF FF FF FF FF FF FF FF 20 10 47 48
[KA8] 26: 49 F1 4B FF 4D F1 4F 50 51 52 53 F6 FF FF FF FF
[CK7] 27: 1B 78 79 7A 7B 7C 7D 7E 7F 80 81 82 83 08 09 10
[OH7] 28: 11 12 13 14 15 16 17 18 19 5B 5D 0D 04 1E 1F 20
[WR7] 29: 21 22 23 24 25 26 7B 7D 5C 02 23 2C 2D 2E 2F 30
[SV7] 30: 31 32 2C 2E 2D 01 2A 08 20 40 68 69 6A 6B 6C 6D
[6W7] 31: 6E 6F 70 71 20 10 47 48 49 F1 4B FF 4D F1 4F 50
[K98] 32: 51 52 53 F6 FF FF FF FF 1B 31 AB 33 AC 35 36 37
[V98] 33: EC 39 F7 E1 AF 08 09 71 77 90 D5 D1 B8 75 F5 EA
[P08] 34: 14 5B 5D 0D 04 E0 E4 7F C6 D8 B5 6A 6B 9C 40 40
[G28] 35: 7E 02 23 9D 78 80 4A CF BE 6D 2C 2E 2D 01 2A 08
[DK7] 36: 20 40 3B 3C 3D 3E 3F 40 41 42 43 44 20 10 47 48
[KA8] 37: 49 F1 4B FF 4D F1 4F 50 51 52 53 F6 FF FF FF FF
[A08] 38: 1B 31 FD 33 AC 35 36 37 38 39 F3 E1 A9 08 09 71
[S38] 39: 77 8A DA C2 BF C4 69 A7 E8 7B 7D 0D 04 A6 73 64
[238] 40: C3 C5 B4 B3 6B 9C 5B 5D 7C 02 23 9D 78 63 C0 C1
[T28] 41: D9 6D 2C 2E 2D 01 2A FF 20 40 3B 3C 3D 3E 3F 40
[WU7] 42: 41 42 43 44 20 10 47 48 49 FF 4B FF 4D F1 4F 50
[NA8] 43: 51 52 53 F6 FF FF FF FF 1B 31 FD 33 AC 35 36 37
[568] 44: 38 39 F2 E1 AA 08 09 71 77 8A D6 D2 B7 75 69 A7
[P38] 45: E8 7B 7D 0D 04 A6 73 64 C7 D7 B6 6A 6B 9C 5B 5D
[V48] 46: 7C 02 23 9D 78 63 D3 D0 BD 6D 2C 2E 2D 01 2A FF
[DK7] 47: 20 40 3B 3C 3D 3E 3F 40 41 42 43 44 20 10 47 48
[GE8] 48: 49 FF 4B FF 4D FF 4F 50 51 52 53 F6 FF FF FF FF
[D28] 49: FF 78 79 7A 7B 7C 7D 7E 7F 80 81 82 83 FF FF 10
[V27] 50: 11 12 13 14 15 16 17 18 19 FF FF FF 1E 1F 20
[088] 51: 21 22 23 24 25 26 FF FF FF FF FF 2C 2D 2E 2F 30
[0N8] 52: 31 32 FF FF FF FF FF FF 20 FF 68 69 FF FF FF FF
[849] 53: 6E 6F 70 71 FF FF FF FF FF FF FF FF FF FF FF FF
[GE8] 54: FF FF FF FF FF FF FF FF 27 60 5E 91 91 86 92 92
[VV7] 55: 8F A0 85 83 86 A6 8F 87 87 80 80 80 82 8A 88
[F18] 56: 90 90 90 A1 8D 8C A4 A4 A4 A5 A5 A5 A2 95 93 A3
[OV7] 57: 97 96 89 8B 98 07 1F 2E FF 2E 08 01 1E 06 60
[V38] 58: B8 40 00 8E D8 2E FF 0E 4E 01 75 E9 2E 80 3E 4B
[M27] 59: 01 FF 74 12 2E C6 06 4B 01 FF 2E 8B 16 12 01 B0
[628] 60: 0E EE 42 33 C0 EE 2E A1 4C 01 2E A3 4E 01 EB C5
[WN7] 61: 61 07 1F 2E FF 2E 04 01 2E 8B 16 12 01 83 C2 06
[048] 62: EC A8 80 74 FB EC A8 80 75 FB 42 32 C0 EE 42 B8
[DW7] 63: B0 00 23 C2 3C B0 75 03 83 EA 03 EE 40 10 2E 8B
[PX7] 64: 16 12 01 EE 42 EC 4A 8A F8 B0 11 EE 42 EC 4A 8A
[8Y7] 65: D8 81 FB 8D 0B B4 19 72 02 B4 57 B0 06 EE 8A C4
[RS7] 66: 42 EE C3 FA 08 26 97 00 E9 E0 01 F6 06 18 00 04
[7M7] 67: 75 F6 80 0E 18 00 04 32 C0 EB 08 90 80 26 18 00
[ZW7] 68: FB B0 01 B4 85 50 E8 51 01 E8 DB 04 58 FB CD 15
[WY7] 69: E9 C4 01 FB FC 1E 06 B8 40 00 8E D8 8E CD 04
[BV7] 70: 2E 80 3E 4B 01 00 74 09 E8 6D FF 2E C6 06 4B 01
[PV7] 71: 00 2E A1 4C 01 2E A3 4E 01 E4 60 B4 10 3C FA 74
[LD8] 72: A2 02 E4 3C FE 74 9C 3C 69 74 A0 3C E9 74 AD A8
[DP7] 73: 80 74 08 E8 55 04 72 0B E9 70 01 E8 4D 04 72 29
[GK7] 74: E9 32 04 57 BF 18 00 F6 C3 70 75 03 BF 17 00 80
[MY7] 75: F3 FF 20 1D 5F 3C 75 39 33 C0 A0 19 00 88 26
[DY7] 76: 19 00 0A C0 74 2C E9 3D 01 F6 C3 F0 74 20 F6 C3
[WP7] 77: 20 75 22 F6 C3 10 75 27 84 1E 18 00 75 14 08 1E
[AU7] 78: 18 00 30 1E 17 00 F6 C3 80 74 1E E9 E7 03 08 1E
[MS7] 79: 17 00 E9 16 01 F6 06 17 00 04 74 DC E9 74 03 F6
[VU7] 80: 06 17 00 04 74 D2 E9 8F 03 FA 50 51 F6 06 97 00
[6S7] 81: 40 75 44 A0 17 00 C0 D8 04 24 07 8A C8 32 06 97
[EH7] 82: 00 24 07 74 32 FA 08 26 97 00 F8 08 0E 97 00 80
[1X7] 83: 0E 97 00 40 E8 63 00 B0 ED E8 26 00 0A F6 06 97
[EY7] 84: 00 80 75 0D 8A C1 E8 19 00 FA F6 06 97 00 80 74
[L48] 85: 06 B0 F4 E8 0C 00 FA 80 26 97 00 3F 59 58 FB E9
[R08] 86: A9 00 50 51 52 8A E0 B2 05 FA E4 64 A8 02 75 F9
[AY7] 87: 8A C4 E6 60 80 26 97 00 4F B9 00 1B FB F6 06 97
[WM7] 88: 00 10 75 12 F6 06 97 00 20 75 02 E2 F0 80 0E 97
[T38] 89: 00 80 FE CA 75 D3 5A 59 58 C3 B0 20 E6 20 C3 8B
[2T7] 90: 1E 1C 00 8B F3 43 43 3B 1E 82 00 75 04 8B 1E 80
[FP7] 91: 00 3B 1E 1A 00 74 08 89 04 89 1E 1C 00 F8 C3 E8
[YZ7] 92: D8 FB B0 B6 E6 43 50 58 B8 00 04 E6 42 86 E0 E6
[XW7] 93: 42 E4 61 50 0C 03 E6 61 B9 00 80 E2 FE 58 E6 61
[Z28] 94: E8 44 03 F9 C3 FE C8 2E D7 3C FF 74 1E 8A E0 32

```

```

[658] 95: C0 EB 13 90 8A E0 FE C8 2E D7 3C FF 74 0D E8 34
[K08] 96: 00 72 08 E8 16 00 E8 96 FF 72 0C FA E8 8B FF E8
[5T7] 97: 15 03 B8 12 90 CD 15 FA 61 07 1F CF 80 FC 15 74
[T38] 98: 06 80 FC 2C 74 08 C3 A8 5A 74 FB B4 2C C3 A8 59
[3T7] 99: 74 F4 B4 15 C3 2E 80 3E 49 01 00 74 19 E8 2F 00
[CK7] 100: 73 26 50 2E A0 49 01 E8 55 FF 58 2E C6 06 49 01
[HR7] 101: 00 72 13 EB 18 90 3C 27 74 08 3C 60 74 04 3C 5E
[X07] 102: 75 0B 2E A2 49 01 F9 C3 BB 68 04 2E D7 F8 C3 53
[AZ7] 103: B7 03 B3 00 3C 20 74 54 02 DF 3C 84 74 4E 02 DF
[PV7] 104: 3C 8E 74 48 02 DF 3C 61 74 42 02 DF 3C 41 74 3C
[4W7] 105: 02 DF 3C 63 74 36 02 DF 3C 43 74 30 02 DF 3C 65
[H08] 106: 74 2A 02 DF 3C 45 74 24 02 DF 3C 69 74 1E 02 DF
[ZY7] 107: 3C 6E 74 18 02 DF 3C 4E 74 12 02 DF 3C 6F 74 0C
[H28] 108: 02 DF 3C 75 74 06 02 DF 3C 79 75 1F 8A C3 2E 80
[PM7] 109: 3E 49 01 27 74 0C FE C0 2E 80 3E 49 01 60 74 02
[218] 110: FE C0 5B 2E C6 06 49 01 00 F8 C3 5B F9 C3 F6 06
[9D7] 111: 17 00 0C 74 41 F6 06 17 00 04 74 07 F6 06 17 00
[7P7] 112: 08 75 03 E9 05 FF E9 C7 FC F6 06 17 00 20 74 03
[9Z7] 113: EB 43 90 EB 6C 90 3C 3B 72 79 3C 44 76 4E 3C 57
[MS7] 114: 74 4A 3C 58 74 46 3C 4A 74 31 3C 4E 74 65 3C 53
[MG7] 115: 74 BC 3C 54 74 5D F6 06 17 00 08 74 03 E9 70 01
[HH7] 116: F6 06 17 00 04 75 40 F6 06 17 00 03 75 B8 F6 06
[9P7] 117: 17 00 20 75 6C BB 50 01 E9 8A FE F6 06 17 00 08
[3U7] 118: 74 31 F6 06 17 00 04 74 2A E9 EE 01 F6 06 17 00
[LL7] 119: 08 75 1A F6 06 17 00 04 75 0D F6 06 17 00 03 74
[8A8] 120: D4 BB A8 01 E9 5E FE BB 00 02 E9 58 FE BB 10 04
[2W7] 121: E9 52 FE 3C 37 75 03 E9 A2 00 F6 06 17 00 08 75
[VR7] 122: 2C F6 06 17 00 04 75 1F F6 06 17 00 03 75 F7 F6
[5K7] 123: 06 17 00 40 74 05 E8 83 01 72 06 BB 50 01 E9 33
[6C8] 124: FE BB A8 01 E9 2D FE BB 00 02 E9 27 FE 3B 06 17
[8N7] 125: 00 04 75 47 F6 06 17 00 03 75 3A 50 3C 3B 72 0C
[R08] 126: 3C 44 76 23 C5 37 74 1F 3C 58 74 1B BB 50 01 8A
[068] 127: E0 FE C8 2E D7 3C 30 72 15 3C 39 76 0A 24 FD 3C
[127] 128: 41 72 0B 3C 5A 77 07 58 BB B0 02 E9 D7 FD 58 BB
[508] 129: 58 02 E9 DF FD BB 08 03 E9 D9 FD F6 06 17 00 03
[D38] 130: 74 06 BB 6D 0F E9 CC FD BB B8 03 E9 C6 FD F6 06
[RM7] 131: 17 00 40 74 8C E8 04 01 72 81 EB 85 F6 06 17 00
[ES7] 132: 03 75 0E F6 06 17 00 04 75 15 B0 2A B4 37 E9 B5
[G28] 133: FD F6 06 17 00 04 75 0E B0 2F B4 37 E9 A7 FD B0
[M48] 134: 00 B4 72 E9 A0 FD E8 0E 00 E8 2E FD 5D CD 05 5D
[T18] 135: E9 A4 FD 80 0E 18 00 08 E8 AC 00 E8 1C FD 80 3E
[WN7] 136: 49 00 74 74 09 2E 8B 16 01 01 A0 65 00 EE F6 06
[9Z7] 137: 18 00 08 75 F9 E9 7F FD A1 80 00 A3 1C 00 A3 1A
[W27] 138: 00 C6 06 71 00 80 E8 7E 00 CD 1B 33 C0 E9 56 FD
[C38] 139: 50 FE C8 BB A8 01 2E D7 2C 30 72 15 3C 09 77 11
[L28] 140: 5B 8A D8 A0 19 00 6B C0 0A 02 C3 A2 19 00 E9 3A
[E68] 141: FD 58 E9 F8 FE F6 06 18 00 08 75 03 E9 37 FE 3C
[V77] 142: 45 74 05 80 26 18 00 07 E9 20 FD 24 7F 3C 38 74
[DR7] 143: 1E 3C 1D 74 1A 3C 2A 74 16 3C 36 74 12 3C 52 74
[KW7] 144: 1B 3C 45 74 0A 3C 46 74 06 3C 3A 74 02 F8 C3 BB
[V08] 145: 50 01 50 FE C8 2E D7 7A D8 58 F9 C3 F6 06 17 00
[PA8] 146: 20 75 EA B3 80 EB F3 FA B4 AE EB 04 90 FA B4 AD
[TU7] 147: E4 64 A8 02 75 FA 8A C4 E6 64 FB C3 50 BB 50 01
[327] 148: FE C8 2E D7 3C 84 74 0F 3C 94 74 0B 3C 81 74 07
[PY7] 149: 3C 61 72 03 3C 7B F5 F5 58 C3 50 51 55 33 C9 B5
[PM7] 150: 10 F6 06 17 00 03 75 13 3C 02 E8 8E 46 01 3B
[X18] 151: 0E 60 00 8A EC 74 0A 8A E9 FE CD B4 01 CD 10 5D
[H48] 152: 59 58 E9 8E 0E 1F B4 B5 B0 09 CD 21 2E 89 1E
[KP7] 153: 04 01 2E 8C 06 06 01 B4 35 B0 1C CD 21 2E 89 1E
[DT7] 154: 08 01 2E 8C 06 0A 01 2E 8E 06 06 01 BF 16 01 8B
[338] 155: F7 B9 2A 00 FC F3 A6 75 03 E9 CA 00 2E C6 06 4A
[NG7] 156: 01 00 BB 80 00 8A 07 25 FF 00 75 03 E9 B0 00 02
[U77] 157: D8 8A 07 2C 30 72 F5 74 19 3C 09 77 EF 32 E4 40
[UU7] 158: 2E C6 06 4A 01 FF 69 C0 66 06 2E A3 4C 01 2E A3
[OF7] 159: 4E 01 2E C6 06 0F 01 03 B8 40 00 8E C0 BF 63 00
[XK7] 160: 26 8B 15 2E 89 16 12 01 83 C2 04 2E 89 16 10 01
[E38] 161: 2E C7 06 46 01 0D C8 B0 0C 8E C0 BF 1E 00 BE
[8Y7] 162: 0C 01 B9 03 00 FC F3 A6 74 2B 2E FE 0E 0F 01 C7
[LX7] 163: 06 60 00 07 06 84 0F CD 10 3C 07 75 18 2E FE 0E
[VU7] 164: 0F 01 2E FE 0E 0F 01 2E C7 06 46 01 0D C0 2E 81
[XY7] 165: 2E 14 01 00 08 2E 8B 0E 46 01 B4 01 CD 10 0E 1F
[P28] 166: B4 25 B0 09 BA 53 05 CD 21 2E 80 0E 4A 01 00 74
[P08] 167: 09 B4 25 B0 1C BA 9D 04 CD 21 BA 86 0A CD 27 BA
[678] 168: 95 0B B4 09 CD 21 BA DB 0B B4 09 CD 21 B2 07 B4
[FU7] 169: 02 CD 21 CD 20 50 61 72 61 6D 65 74 65 72 61 6E
[CL7] 170: 67 61 62 65 20 69 6D 20 54 61 73 74 61 74 75 72
[DR7] 171: 74 72 65 69 62 65 72 20 66 61 6C 73 63 68 2E 0D
[4S7] 172: 0A 41 75 66 72 75 66 3A 20 AE 54 33 20 6E AF 20
[528] 173: 6E EE 7B 30 2E 2E 39 7D 0D 0A 24 4E 69 63 68 74
[9S7] 174: 20 28 65 72 6E 65 75 74 29 20 69 6E 73 74 61 6C
[C24] 175: 6C 69 65 72 74 21 0D 0A 24

```

Listing 2. »t3.hex« können Sie in Scheck eingeben

Hyperkey Plus

Tastatur Total

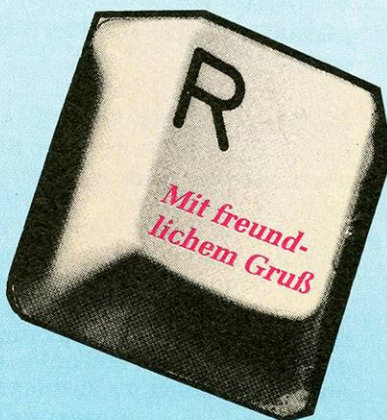
Der speicherresidente Tastatur-Manager **Hyperkey** bringt Ihrer Tastatur das Zaubern bei: Sie können sämtliche Tasten ihres Computers mit neuen Zeichenfolgen belegen.

Vorbei sind die Zeiten, wo Sie zum Wechsel zwischen Ihrer Textverarbeitung und der Tabellen-Kalkulation endlose Befehlsfolgen eingeben mußten – **Hyperkey** erledigt dies mit einem Tastendruck!

Ihre Textverarbeitung wird zum Grafik-Editor mittels der mitgelieferten Blockgrafik-Tastaturbelegungen.

Enorme Arbeitserleichterung für Programmierer: Optimales Turnaround auf Knopfdruck. Oder legen Sie die Strukturen Ihrer Lieblings-Sprache auf bestimmte Tasten. Mit <Alt> + P eine ganze Programmschablone! Belegungen für Pascal und Basic im Lieferumfang.

- Help-Window auf Tastendruck.
- Alle Tastenbelegungen in Übersicht in einem eigenen Fenster.
- Jede Tastenbelegung besitzt einen Kommentar zur Wiedererkennung.
- Jederzeit Speichern und Lesen der Belegungen in Dateien.
- Komfortables, menügesteuertes Installationsprogramm für Farben, Datei- und Macro-Voreinstellungen.
- Fatale Fehler werden abgefangen: z.B. Schreib-/Lese-Fehler oder offener Diskettenschacht.
- **Hyperkey** ist aus allen Programmen heraus aufrufbar.



- Tastatur-Macros können jederzeit mit einem Tastendruck aktiviert oder deaktiviert werden.
- Sämtliche Steuerzeichen wie zum Beispiel <Return> oder <Cursor links> in den Belegungen verwendbar.
- Tastenbelegungen können sich auch gegenseitig aufrufen (Chaining).
- **Hyperkey** kann auch über eine Interrupt-Schnittstelle von selbstgeschriebenen Programmen aus angesprochen werden.
- "Abspielen" von Tastatur-Macros, ohne daß eine Taste gedrückt wird.
- Ein Handling der Scan-Codes ist möglich.
- Funktionstastenbelegung auch an den ANSI-Treiber übergebbar.

Kostenlos im Lieferumfang:

TSR-Verwaltung – Speicherresidente Programme im Griff. Keine Probleme mehr mit residenten Programmen! Zählung der residenten Speicherplatzfresser. 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, Entfernen bei Speicherplatzmangel oder Unverträglichkeiten gemäß den gesetzten Markierungen, unbedenkliches Neueinladen entfernter Programme.

Hyperkey Plus wird inklusive umfangreicher deutscher Dokumentation geliefert.

Notwendige Gerätekonfiguration:
PC XT/AT mit 256kB oder mehr, 1 Diskettenlaufwerk
bzw. Festplatte. MS-DOS 2.0 oder größer. Alle marktüblichen Grafikkarten werden unterstützt.

© 1988 DMV

Best.-Nr. 229

99, – DM

unverbindliche Preisempfehlung

DMV Verlag · Fuldaer Str. 6 · 3440 Eschwege

Disketten unter der Lupe

Es schadet nie, wenn man als Diskettenbenutzer einiges über deren Aufbau weiß. Vermeintliche Katastrophen, etwa versehentliches Löschen von Dateien, verlieren dann schnell ihren Schrecken.

5 1/4 Zoll Disketten mit 360 KByte verfügbarer Speicherkapazität sind heute am weitesten verbreitet. Im folgenden beziehen sich deshalb alle Angaben ohne besonderen Vermerk auf dieses Speichermedium. 360 KByte bedeutet 360 x 1024 Byte. Die Definition der Kapazität einer Diskette erfolgt bei der Formatierung durch das Betriebssystem. Die Oberfläche der Diskette wird dabei in 40 konzentrische Kreise, sogenannte »Tracks« (Spuren) unterteilt. Jede dieser Spuren wiederum besteht aus neun Sektoren, und jeder Sektor enthält 512 Byte.

Eine Diskette läßt sich nun beidseitig nutzen. Dies ergibt einen Gesamtspeicherplatz von 512x9x40x2, also 368640 Byte. Davon werden jedoch noch die zwölf Sektoren, die zur Organisation der Diskette nötig sind, abgezogen.

368640 Byte - 12 x 512 Byte =
362496 Byte.

Die einzelnen Sektoren sind nummeriert. Die Numerierung beginnt auf Seite 0, Spur 0 mit Sektor 0 bis 8 und geht weiter auf Seite 1, Spur 0 mit Sektor 9 bis 17. Nun folgt Seite 0, Spur 1 mit Sektor 18 bis 26 etc.

Der Inhalt der ersten Sektoren ist bekannt: Sektor 0 ist reserviert und zeigt, daß es sich um eine MS-DOS-Diskette handelt. Die Sektoren 1 und 2 enthalten den FAT (siehe Tabelle 2), die Sektoren 3 und 4 eine Kopie des FAT. In den Sektoren 5 bis 11 (hexadezimal »B«) steht das Directory.

Die kleinste Speichereinheit, in die das MS-DOS eine Datei schreiben kann, ist ein Cluster. Eine Datei umfaßt deshalb meist eine größere Anzahl von Clustern. Ein Cluster besteht aus zwei Sektoren (1024 Byte). Praktisch bedeutet das, daß selbst für eine ganz kleine Datei immer mindestens 1024 Byte auf der Diskette reserviert werden müssen. Die Cluster sind etwa wie die Sektoren nummeriert. Da die ersten Sektoren durch die FAT und das Directory belegt sind, beginnt die Numerierung nicht bei Sektor 0. Die Nummer des ersten Clusters einer Datei befindet sich im Directory (Bild) und damit ist der Dateibeginn bekannt. Mit Hilfe der

Eintragung im FAT ist es kein Problem, den jeweils nächsten Cluster zu finden. Dieser zeigt wiederum auf den nächsten, und so fort.

Mit dem Hilfsprogramm »Debug« und einiger Kenntnis über den Disketten- aufbau, können Sie vieles entdecken

Um Änderungen an der Diskette durchzuführen, formatieren Sie eine neue Diskette mit dem Befehl »format/s«. Um die Änderungen zu testen, sollten Sie alle Dateien, die zum Start des Systems nötig sind, sowie das Hilfsprogramm »debug.com« auf diese Diskette kopieren. Es ist für diese Versuche nützlich, außerdem noch einige Textdateien zu kopieren und ein Unterverzeichnis anzulegen.

Um das Verzeichnis der Diskette zu lesen, arbeiten Sie mit dem Programm »Debug«. Das Directory befindet sich in den Sektoren 5 bis B (hexadezimal). Debug lädt diesen Bereich mit »L CS:0100 0 5 7«, danach werden von Laufwerk »0«, ab Sektor 5 sieben Sektoren in den Speicher geladen. Der Speicherinhalt wird mit »D CS:0100 24F« angezeigt. Mit Debug lassen sich die Daten im Speicher modifizieren; der dazugehörige Befehl ist »E (Adresse) (Daten)« (alle Eingaben hexadezimal). Hierzu einige Beispiele:

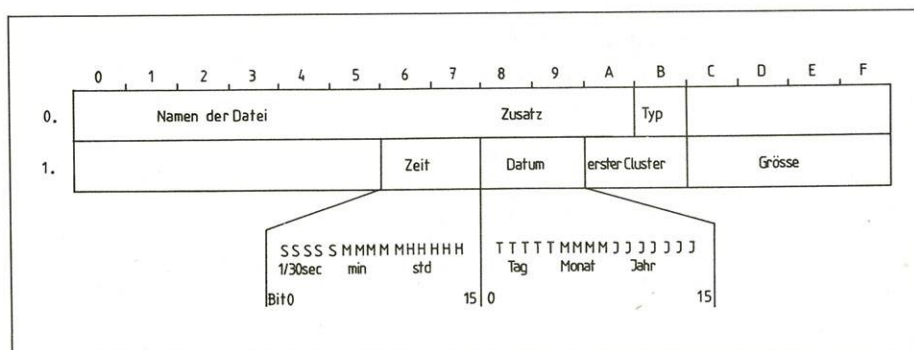
- Die Befehle »E CS:010B 21« und »E CS:012B 21« machen beispielsweise die versteckten Dateien »ibmbio.com« und »ibm.dos« sichtbar.

- Das Datum von »command.com« läßt sich mit »E CS:0158 41 10« in »5. Januar 1988« ändern (Bild). Mit dem Befehl »W CS:0100 0 5 7« übertragen Sie die Änderungen wieder auf die Diskette. Wenn Ihre Diskette mit »format/s« formatiert wurde, können Sie die Befehle in der oben beschriebenen Reihenfolge ausführen.

Nun zu einem anderen Thema. Wenn Sie Dateien einmal aus Versehen gelöscht haben, sind sie noch nicht verloren. Denn alle Daten existieren solange, bis sie mit einer anderen Datei überschrieben werden. Damit der Computer aber weiß, welche Dateien als gelöscht gelten, wird der erste Buchstaben des Dateinamens im Directory einfach durch »E5« ersetzt. Wenn Sie diesen Eintrag durch einen Buchstaben (14hex bis 5Ahex) ersetzen, erscheint die Datei wieder im Verzeichnis. Freuen Sie sich aber nicht zu früh: Kurze Dateien lassen sich zwar lesen, aber sie werden von der als nächstes gespeicherten Datei überschrieben. Um eine kurze Datei zu retten, genügt es jedoch, den Dateinamen im Directory wieder herzustellen und sofort auf eine andere Diskette zu kopieren.

Damit der Computer weiß, wo er die Dateien suchen muß, findet sich in den Sektoren 1 bis 4 eine Tabelle, die FAT (File Allocation Table). Tatsächlich existiert diese Tabelle sogar zweimal – einmal in den Sektoren 1 und 2 und ein zweitesmal in den Sektoren 3 und 4. Jeder Eintrag ist 3x4 Bit groß. Für nicht benutzte oder gelöschte Cluster ist dieser Eintrag »000«. Bei existierenden Dateien zeigt dieser Eintrag auf den nächsten relativen Cluster (Tabelle 2). Jeder FAT beginnt mit einem Code, der das verwendete Diskettenformat definiert:

- »F8« für eine Harddisk,
- »FC« für eine einseitige Diskette mit neun Sektoren pro Spur (180 KByte)



Ein typischer Directory-Eintrag

- »FD« für eine doppelseitige Diskette mit neun Sektoren pro Spur (360 KByte)
- »FE« für eine einseitige Diskette mit acht Sektoren pro Spur (160 KByte),
- »FF« für eine zweiseitige Diskette mit acht Sektoren pro Spur (320 KByte). Ein nicht lesbarer Cluster ist mit »FF7« markiert.

Wenn Sie sich näher mit der Struktur einer Diskette und deren Inhalt beschäftigen, hat es meist den Grund, eine verlorene Datei zu retten. Zu diesem Zweck fertigen Sie als erstes mit »diskcopy« eine Kopie der Diskette an. Der Befehl »diskcopy« kopiert alle Sektorinhalte genau identisch auf die andere Diskette. Das schließt also auch die als gelöscht gekennzeichneten Dateien mit ein. Nun kann der Name der Datei im Directory wiederhergestellt werden. Bei längeren Dateien und um eine Datei dauerhaft zu retten, müssen Sie die Einträge in der FAT ebenfalls wiederherstellen. Dies läßt sich zwar »von Hand« erledigen, ist aber recht mühsam. Die Darstellung des FAT durch Debug eignet sich dazu nicht besonders (Tabelle 2), besser geht es mit entsprechenden Hilfsprogrammen (zum Beispiel »Norton Utilities« oder »PC Tools«).

Zum Restaurieren einer Diskette ist es des öfteren notwendig, die Clusternummer in die Sektornummer umzurechnen, und dazu gehen Sie nach einem bestimmten Schema vor. Angenommen der Cluster hat die Nummer 53 (hex). Dann zählen Sie dazu den Offset ($53 + 4 = 57$). Jeder Cluster besteht aus zwei Sektoren ($57 \times 2 = AE$). Die gesuchten Daten befinden sich also in den Sektoren AE und AF. Sie können mit Debug gelesen, angezeigt und geändert werden.

Wie findet man nun die Einträge, die sich nicht im Hauptverzeichnis befinden? Die Nummer des ersten Clusters des Unterverzeichnisses kennen Sie aus dem Hauptverzeichnis. Diese Nummer rechnen Sie in die absolute Sektornummer um und zeigen sie mit Debug an. Ist das Directory größer, müssen Sie die Nummern der Cluster im FAT suchen.

Wenn Sie nun für das Programm Debug gar keine Sympathien entwickeln können: Es gibt bei PC-SIG unter der Nummer 598 die Diskette »Disk-Tool V1.6c«. Ob Sie mit Debug oder Disk-Tool arbeiten, wichtig ist, daß Sie vor dem nächsten Datenverlust über das Innenleben Ihrer Disketten Bescheid wissen.

(Johannes Baumann/ue)

Diskettenformate von 5 1/4 Zoll-Disketten unter IBM-DOS

Kapazität (Byte)	160K	180K	320K	360K	1.2M
Spuren (Anzahl)	40	40	40	40	80
Sektoren/Spur	8	9	8	9	15
Byte/Sektor	512	512	512	512	512
Seiten	1	1	2	2	2
Sektoren/Cluster	1	1	2	2	2
Reservierter Sektor Nr.	0	0	0	0	0
FAT 1 (Sektor Nr.)	1	1+2	1+2	1+2	1-7
FAT 2 (Sektor Nr.)	2	3+4	3+4	3+4	8-E
Directory (Sektor Nr.)	3-6	5-8	*	5-B	F-1D
Einträge (max.)	64	64	112	112	224

* Die Sektoren der 320 KByte Disketten wurden unter DOS 1.1 anders nummeriert (zuerst Seite 0, danach Seite 1). Deshalb befindet sich das Directory in den Sektoren 5-7 und 320 + 321 (entsprechend 5-9).

Tabelle 1. Diskettenformate von 5 1/4 Zoll-Disketten unter IBM-DOS

File Allocation Table

Beispiel der ersten Eintragungen in einem FAT:

Eintrag	FDF	FFF	003	004	005	006	007	008	009	00A	00B					
für Cluster			2	3	4	5	6	7	8	9	A					
und wie Debug sie zeigt:																
nnn:																
0100	FD	FF	FF	03	40	00	05	60	00	07	80	00	09	A0	00	0B
Um diese Darstellung zu lesen, muß man die Zeile umkehren,																
0B	00	A0	09	00	80	07	00	60	05	00	40	03	FF	FF	FD	
und Dreiergruppen bilden:																
0B	00A	009	008	007	006	005	004	003	FFF	FFD						
	9	8	7	6	5	4	3	2								

Die Datei ist meistens an einem Stück aufgezeichnet, weil es am Anfang einer Diskette normalerweise keine defekten Cluster gibt, da die äußeren Spuren länger sind als die inneren. Viele Disketten beginnen mit dem System oder einer längeren Datei. Es wäre jedoch ebenso möglich, daß beispielsweise Eintrag 6 »009« ist und Cluster 7 und 8 durch eine andere, ältere Datei belegt sind.

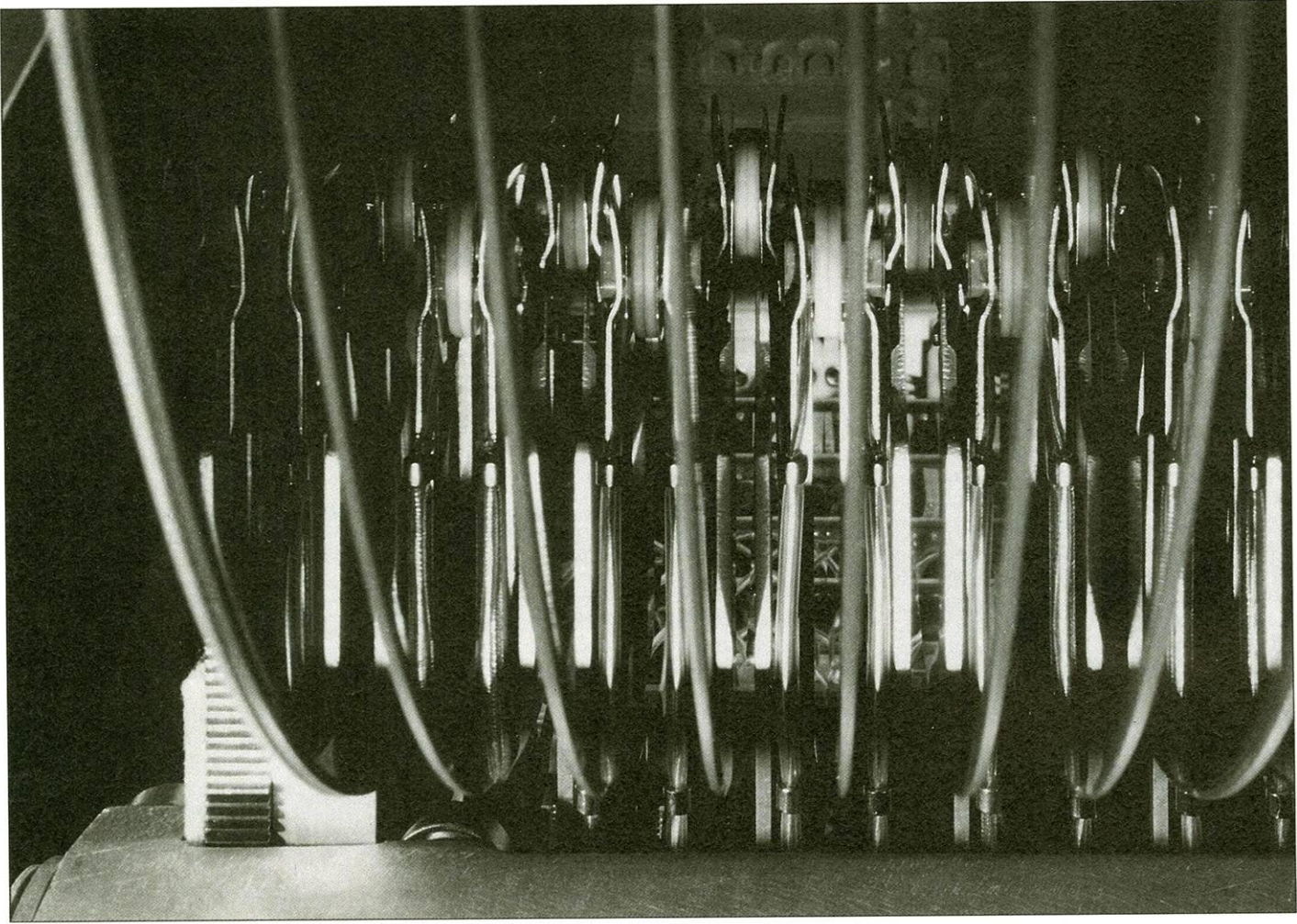
Tabelle 2. So liest man den FAT mit Debug

Dateitypen

In der ersten Zeile jedes Directory-Eintrags (an Adresse B) steht eine Zahl, die den Typ der Datei definiert:

01	Datei kann nur gelesen werden
02	Versteckte Datei
04	Systemprogramm
08	Label
10	Subdirectory
20	Archiv

Tabelle 3. Die verschiedenen Dateitypen



Die Welt der Festplattenspeicher

Der konstante Preisverfall brachte die leistungsfähigen Festplattenlaufwerke auch in für Heimanwender erschwingliche Regionen. Wir erläutern Ihnen den Aufbau dieses interessanten Massenspeichers und klären Sie über die Unterschiede zwischen den angebotenen Modellen auf.

Der »IBM XT« (Abkürzung für »Extended Technology«) war der erste Computer, der serienmäßig mit einer Festplatte ausgerüstet wurde. Die Festplatte vereinfacht den Umgang mit dem Computer erheblich, da sie im Vergleich zum Diskettenlaufwerk viele Vorteile aufweist. Zunächst einmal entfällt das lästige Wechseln der Disketten und die Notwendigkeit der Beschriftung und strukturierten Aufbewahrung. Desweiteren ist eine Festplatte wesentlich schneller als ein Diskettenlaufwerk. Außerdem muß man sich aufgrund der hohen Speicherkapazität selten über Platzprobleme Gedanken machen. Anstelle von »Festplatte« fällt auch

oft der Begriff »Winchester Drive«, der sich auf die Existenz eines Winchester Gewehrs mit dem Kaliber 30 zurückführen läßt, denn die ersten Festplatten von IBM hatten den Namen 3030, da sie zweimal 30 MByte Daten aufnehmen konnten. Grundsätzlich gesehen werden Festplatten in der gleichen Weise angesteuert wie Diskettenlaufwerke. Es gibt allerdings einige mehr oder weniger gravierenden Unterschiede, die vor allem für den Programmierer von Bedeutung sind. Einen wesentlichen Unterschied bemerkt man schon, wenn man einen PC mit eingebauter Festplatte einschaltet: Während die Motoren des

Diskettenlaufwerkes ausschließlich dann in Bewegung gesetzt werden, wenn Daten gelesen oder geschrieben werden sollen, dreht sich die Festplatte permanent, und zwar mit ungefähr 3600 Umdrehungen pro Minute, also ungefähr zwölfmal schneller als eine PC/XT-Diskette und mit der zehnfachen Geschwindigkeit einer AT-Diskette. Das bei der Festplatte verwendete Material ist Aluminium, während Disketten hauptsächlich aus Kunststoff bestehen. Die Aluminiumplatte ist in einem staubfreien Gehäuse untergebracht (siehe Bild).

Die meisten Festplatten besitzen eine Größe von 5 1/4 Zoll, obwohl es auch Festplatten im 3 1/2 Zoll-Format gibt. Entsprechend dem Aufbau einer Diskette werden auch bei einer Festplatte die Daten in konzentrischen Spuren abgelegt. Die geringe Höhe des Schreib-/Lesekopfes über der Platte gewährleistet eine höhere Aufzeichnungsdichte. Auf diese Weise finden bis zu 900 Spuren auf einer Platten-seite Platz. Jede dieser Spuren besteht gewöhnlich aus 17 Sektoren. Ein Festplattenlaufwerk verfügt über bis zu 15 Schreib-/Leseköpfe und die dazugehörigen Platten.

Zylinder	Köpfe	Megabyte
306	4	10
615	4	21
615	6	32
940	8	65
940	6	49
615	4	21
462	8	32
733	5	32
900	15	117
820	3	21
855	5	37
855	7	52
306	8	21
733	7	44

Tabelle 1. Die Struktur der AT-Tabelle

In der Ansteuerung der Festplatte liegt ein wesentlicher Unterschied zwischen PC/XT- und AT-Computern. Während der PC/XT vier unterschiedliche Festplattenformate verwenden kann, ist der AT in der Lage, bis zu 15 verschiedene Größen zu verwalten. Beim PC/XT befinden sich die Informationen über den Aufbau der Festplatte im Controller, während der AT dafür eine spezielle Festplattentabelle im BIOS besitzt. Über diese Parameter im PC/XT/AT wird die Festplatte für die Benutzung durch das BIOS angemeldet. Die AT-Tabelle liegt ab Adresse FE401h im ROM. Tabelle 1 zeigt ihre genaue Struktur.

Theoretisch kann man bis zu 117 MByte auf einer Festplatte speichern

Laut den Angaben der Tabelle lassen sich theoretisch durchaus Festplatten mit einer Kapazität bis zu 117 MByte im AT einsetzen. Das verhindert aber die heutige Speichertechnologie, die diese extreme Schreiddichte nicht realisieren kann. 80 MByte sind zur Zeit das Maximum an Speicherkapazität der Plattenlaufwerke, die im PC/XT/AT Verwendung findet. Das Betriebssystem steuert eine Festplatte normalerweise genauso an wie ein Diskettenlaufwerk. Doch obwohl der Anwender auch bei der Festplatte den Kopf, die Spur und den Sektor spezifizieren muß, gibt es einige Unterschiede. So ist der Benutzer in der Lage, die Festplatte für verschiedene Betriebssysteme einzurichten. Man kann beispielsweise einen Teil der Platte für MS-DOS, eine weiteren für Xenix und einen anderen für CP/M verwenden. Diese Aufteilung der Festplatte in mehrere Bereiche nennt

man »Partition« (englisch für »Aufteilung«).

Fast alle eingesetzten Betriebssysteme benutzen die gleichen BIOS-Interrupts für ihr internes Datenmanagement. Mit dem BIOS können Sie also auf die gesamte Festplatte zugreifen, während die Aktivitäten des MS-DOS auf die Bereiche des Datenträgers begrenzt sind, die auch für MS-DOS eingerichtet sind. Die Informationen, welche Bereiche der Festplatte von welchem Betriebssystem benutzt werden, sind in der »partition table« (englisch für »Aufteilungstabelle«) abgelegt. In dieser Tabelle ist gespeichert, an welcher Stelle welches Betriebssystem abgelegt ist, wie lang der Bereich ist und welcher Bereich aktiviert wurde. Die »partition table« befindet sich im ersten Sektor auf der ersten Spur des ersten Kopfes. Mit dem Betriebssystem kann man anschließend auf die Sektoren der Platte zugreifen, die außerhalb der Tabelle liegen. Ausgelesen werden die Daten über den Betriebssystem-Interrupt 13h.

Vier verschiedene Betriebssysteme können auf einer Festplatte eingesetzt werden

Die Tabelle beginnt bei 1BEh, ist insgesamt 64 Byte lang und besteht aus vier Blöcken zu 16 Byte. Jeder Block

enthält die Informationen eines Betriebssystembereichs, was dazu führt, daß man höchstens vier Betriebssysteme gleichzeitig einsetzen kann. Jeder Eintrag weist eine bestimmte Struktur auf, die Sie in Tabelle 2 sehen.

Es mag Sie überraschen, auf welche Weise die Angaben über den ersten und letzten Sektor in der Tabelle codiert sind. Denn aus welchem Grund werden die oberen beiden Bits in einem anderen Byte abgespeichert, obwohl doch in dem Byte, das auch die unteren sechs Bit enthält, noch zwei Bit unbenutzt sind? Die Antwort ist einfach: Das oben vorgestellte Format entspricht dem vom BIOS für die Adressierung von Sektoren verwendeten Aufbau. Die Daten der »partition table« brauchen Sie also nicht erst umzuwandeln.

Da die »partition table« 64 Byte lang ist und bei 1BEh beginnt, bleiben in dem 512 Byte langen Sektor noch zwei Byte übrig. Sie enthalten die Werte 55h und AAh.

Damit sind wir ans Ende unseres Streifzuges durch die Welt der Festplattenspeicher gelangt, der Ihnen hoffentlich die eine oder andere noch unbekannte Information liefern konnte.

(Burt Roman/ue)

Byte	Funktion
0	Boot-Kennzeichen. Wenn dieses Byte den Wert 80h enthält, wird der angegebene Bereich gebootet. Nur einer der vier Blöcke darf in dieser Weise gekennzeichnet sein.
1	Erste Seite des Bereiches
2	Die unteren sechs Bit geben den ersten Bereichs Sektor an, während die oberen beiden Bit die oberen beiden Bit der Spur darstellen, in der der Bereich beginnt
3	Die unteren sechs Bit sind die unteren sechs Bit der Spur, in der der Bereich beginnt
4	FAT-Kennzeichen (Abkürzung für »File Allocation Table«). Dieser Wert gibt an, welches FAT-Format das spezifizierte Betriebssystem benutzt. Beträgt der Wert 4, so handelt es sich um einen 16-Bit-FAT, ein Inhalt von 1 weist auf einen 12-Bit-FAT hin, während der Wert 0 Fremdbetriebssysteme kennzeichnet
5	Letzte Seite des Bereiches
6	Die unteren sechs Bit geben den letzten Bereichs Sektor an, während die oberen beiden Bit die oberen beiden Bit der Spur darstellen, in der der Bereich endet
7	Die unteren sechs Bit sind die unteren sechs Bit der Spur, in der der Bereich endet
8-11	Langwort im Intel-Format (zuerst das niedrig- und darauf folgend das höherwertige Byte), das die Anzahl der Sektoren angibt, welche dem Betriebssystem vorausgehen
15-16	Langwort im Intel-Format, das die Anzahl der Sektoren des Betriebssystems bestimmt

Tabelle 2. Jeder Eintrag in die »partition table« weist diese Struktur auf



Der PC – glasklar

Bei diesem Beitrag handelt es sich um eine überarbeitete und aktualisierte Version des großen PC-Kurses gleichen Titels, der in den vergriffenen Ausgaben 8'87 bis 1'88 der DOS International veröffentlicht war. Dieser Kurs brachte Licht ins Dunkel der PC-Hardware und des Betriebssystems.

Mit der Überarbeitung und Neuauflage dieses Kurses in einem Sonderheft erfüllen wir einen oft geäußerten Wunsch der Leser der DOS International.

Der PC – glasklar (1)

Nur wer seinen PC genau kennt, kann Geschwindigkeit und Leistungsfähigkeit voll nutzen. Dieser Kurs lüftet die Geheimnisse um Betriebssystem und Hardware Ihres Computers und zeigt, wie Sie in Pascal, Basic oder anderen Sprachen das Optimum herausholen können.

Mit diesem Kurs gehen wir den PC von einer bislang wenig beachteten Seite an: Sie finden hier jede Menge Informationen, Tips und Tricks über die interne Arbeitsweise Ihres PC, über Betriebssystem und Hardware. Praktisch alle diese Informationen können Sie sofort bei der Programmierung in Basic, Pascal oder einer anderen Sprache umsetzen.

Beginnen wir mit einigen Grundlagen: Alle IBM-kompatiblen PCs haben grundsätzlich die gleiche Hardware-Architektur. Als Prozessoren werden die Intel-Chips 8088, 8086 und 80286 verwendet. Der 8088 und der 8086 sind intern völlig identisch aufgebaut. Sie unterscheiden sich nur durch die Breite des Datenbusses. Der 8088 verfügt über einen 8-Bit-Datenbus, das heißt, er kann mit einem Speicherzugriff nur jeweils ein Byte aus dem Speicher lesen. Intern jedoch kann er wie die beiden anderen Chips Daten 16-Bit-

weise manipulieren. Der 8086 ist dagegen ein echter 16-Bit-Prozessor. Er arbeitet intern wie extern mit 16-Bit-Daten und hat so einen deutlichen Geschwindigkeitsvorteil gegenüber dem 8088. Beide Prozessoren sind in handelsüblichen Geräten der PC- und XT-Klasse zu finden. Die Bezeichnung XT weist übrigens nur auf ein serienmäßig vorhandenes Festplattenlaufwerk hin; die interne Architektur ist sonst völlig identisch mit dem einfachen PC.

Der Intel 80286 schließlich wird in den Geräten der AT-Klasse verwendet. Es handelt sich dabei um einen Prozessor mit interner 32-Bit-Architektur, der aber wie der 8086 nur über einen 16-Bit-Datenbus verfügt. Der 80286 kann 8086-Software problemlos bearbeiten, da die Architektur des Prozessors für eine 8086-Kompatibilität ausgelegt wurde. Wenn Sie einen AT besitzen, können Sie alle Angaben zum normalen

PC und zum 8088/8086-Prozessor unverändert auch für Ihr Gerät verwenden. Hier zeigen sich eben die Vorteile des auf Kompatibilität bedachten PC-Konzepts.

Der 80286 bietet auch einen zusätzlichen, nicht-kompatiblen Betriebsmodus, den sogenannten geschützten Modus (englisch: »protected mode«). Dieser Modus wird zum Beispiel im neuen Betriebssystem OS/2 voll ausgenutzt. Solange Sie aber mit MS-DOS arbeiten, wird dieser Modus nicht eingesetzt, und daher wird hier der geschützte Modus nicht weiter erwähnt.

Obwohl ein Verständnis dieses Kurses keine Assembler-Kenntnisse voraussetzt, ist es ratsam, sich mit dem Aufbau des Prozessors vertraut zu machen. Der 8088/8086 hat 14 interne Register, nämlich vier allgemeine Datenregister, vier Zeiger- beziehungsweise Indexregister, vier Segmentregister, einen Befehlszeiger

Bits, Bytes, Paragraphen

Hier sind einige grundlegende Begriffe erläutert, die im Zusammenhang mit der internen Funktionsweise eines Computers immer wieder auftauchen.

Bit: Ein Bit ist die kleinste Informationseinheit. Es repräsentiert Wahrheitszustände wie ja/nein, ein/aus, 0/1 oder richtig/falsch.

Byte: Jeweils 8 Bits werden zu einem Byte zusammengefaßt. In einem Byte lassen sich daher $2 \text{ hoch } 8$, also 256 verschiedene Zustände codieren. Der Zeichensatz Ihres PC ist beispielsweise in einzelnen Bytes codiert.

Wort: Diejenige Anzahl der Bits, die ein Computer intern mit einem Maschinenbefehl bearbeiten kann, wird traditionell als ein Wort bezeichnet. Bei einem PC ist ein Wort immer 16 Bit breit, entspricht also gerade 2 Byte. Weitere übliche Wortbreiten (bei Mini- und Großrechnern) sind 32, 64 oder 80 Bit.

Paragraph: Hier handelt es sich um eine PC-spezifische Einheit. Ein Paragraph sind 16 Byte. Die Grundlage dieser Speichereinheit ist die Tatsache, daß die Segment-Register beim PC den Speicher gerade in 16-Byte-Abschnitten adressieren. Ein Segment kann also immer nur auf einer durch 16 teilbaren Speicheradresse beginnen. Damit kommt den 16-Byte-Schritten beim PC eine derartige Bedeutung zu, daß sich dafür der Begriff Paragraph eingebürgert hat. Er bedeutet eigentlich nichts anderes als »Abschnitt«, wurde jedoch wie andere Begriffe der Computertechnik auch einfach nicht ins Deutsche übersetzt.

Page oder Speicherseite: Üblicherweise bezeichnet man 256 Byte als eine Page oder eine Seite des Speichers (256 Byte Speicher lassen sich durch 8 Bit, also durch ein Byte adressieren). Ab und zu werden auch 512 Byte als eine Page bezeichnet. Dahinter steckt der Gedanke an die 256 16-Bit-Worte, die eine 512-Byte-Page enthält. Es handelt sich dann also quasi um eine »Wort-Seite«.

Segment: Eine PC-spezifische Einheit, die aus der Adressierung des Speichers über Segment-Register resultiert. Ein Speichersegment entspricht 64 KByte.

KByte: Eine Mengenangabe für Speicher. 1 KByte sind 1024 Byte. Die Einheit »K« ist an die Bezeichnung »Kilo« angelehnt, ist aber wegen der besonderen Bedeutung des Binärsystems in der Computertechnik als Zweierpotenz, nämlich als $2 \text{ hoch } 10$ gleich 1024 definiert.

MByte: Ist analog zum KByte eine Mengenangabe für Speicher. 1 MByte (angelehnt an den Begriff »Mega«) sind 1024 KByte.

GByte: Ist analog zu KByte und MByte eine Mengenangabe für Speicher. 1 GByte (angelehnt an den Begriff »Giga«) sind 1024 MByte.

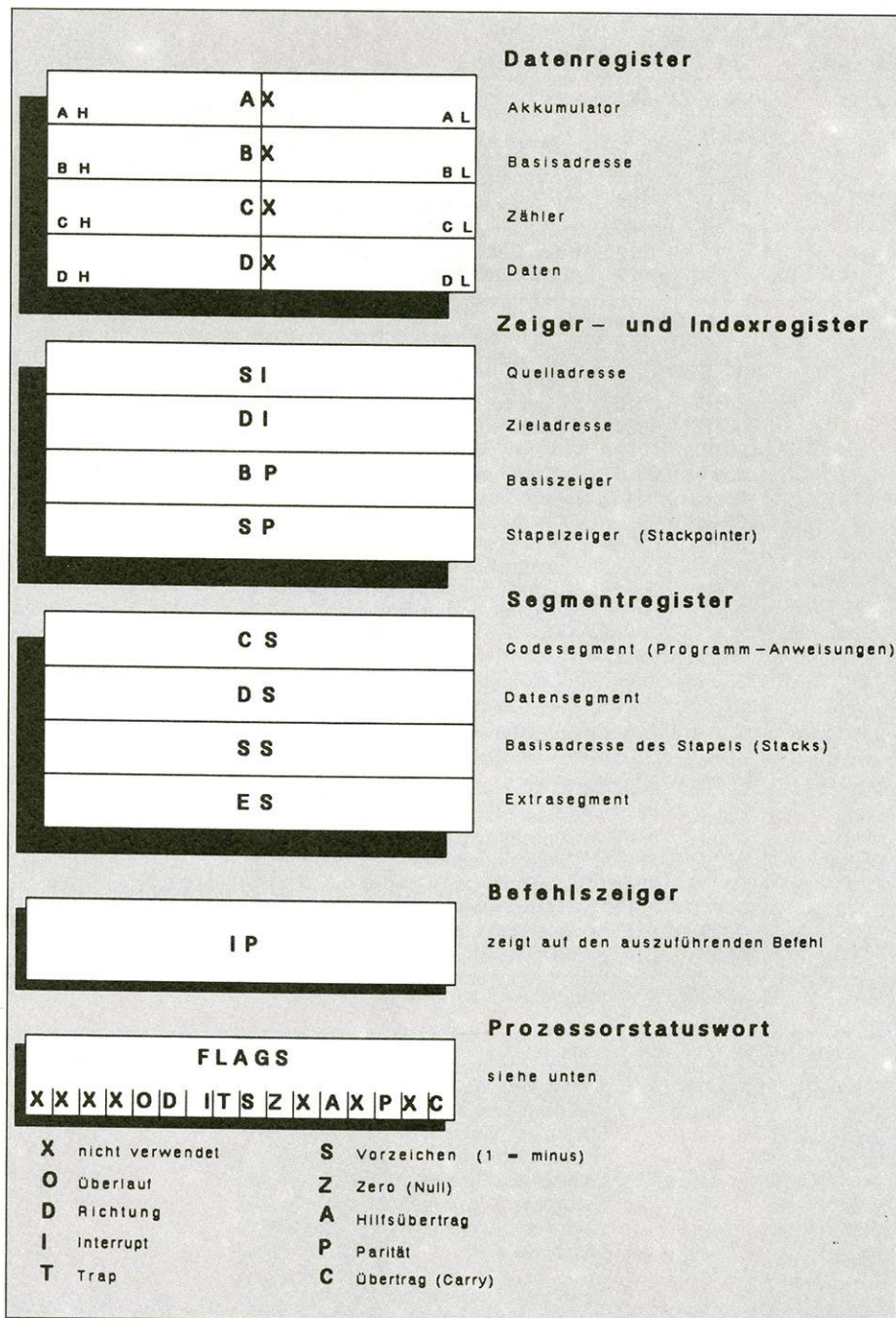


Bild 1. Die Register des 8088/8086

und das Flag-Register (Bild 1). Alle Register sind 16 Bit breit, wie es sich für einen 16-Bit-Prozessor gehört. Im Gegensatz zu früheren 8-Bit-Prozessoren hat der 8086/8088 einige Eigenheiten aufzuweisen, die für die maschinennahe Programmierung auch in einer Hochsprache von immenser Wichtigkeit sind. Sehen wir uns dazu als erstes die Speicheraufteilung an.

Zunächst einmal ist es wichtig zu wissen, daß der 8088/8086 bis zu 1 MByte Speicher direkt adressieren kann. Um diesen Bereich ansprechen zu können, ist ein 20 Bit breiter Adreßbus notwendig (1 MByte entspricht 1024 KByte oder $2 \text{ hoch } 20$ Byte). Da der 8088/8086 aber nur mit

16 Bit umgehen kann, ist ein kleiner Trick notwendig, um diesen 20-Bit-Adreßbus zu realisieren.

Der Trick mit der Segmentierung erlaubt mit 16 Bit die Verwaltung von 1 MByte Speicher

Es wird einfach der gesamte adressierbare Speicherbereich in Segmente zu jeweils 64 KByte eingeteilt. Ein solches Speichersegment von 64 KByte kann gerade mit 16 Bit adressiert werden (64 KByte sind $2 \text{ hoch } 16$ Byte). In 1 MByte Adreßraum passen nun genau 16 Segmente zu 64 KByte, wenn es keine Überlappungen

Segment	Adresse von	bis	Bedeutung
0000	Interrupt-Vektortabelle		
	0000	003F	Hardware-Interrupts
	0040	005F	BIOS Software-Interrupts
	0080	00FF	MS-DOS Software-Interrupts
	0180	019F	für Benutzer reserviert
	01A0	01FF	(nicht verwendet)
	0200	0213	Basic Software-Interrupts
	03C4	03FF	(nicht verwendet)
0040	BIOS-Datenbereich		
	0013	0014	freie Speicherkapazität in KByte
	001A	001B	Zeiger auf Anfang des Tastaturpuffers
	001C	001D	Zeiger auf Ende des Tastaturpuffers
	001E	003D	Tastaturpuffer (16 Tasten zu 2 Byte)
	0049	0049	aktueller Bildschirmmodus
	004A	004B	Bildschirmbreite in Textspalten
	004C	004D	Länge einer Bildschirmseite
	004E	004F	Offsetadresse der aktuellen Bildschirmseite
	0050	005F	Cursorpositionen für acht Bildschirmseiten
	0060	0061	Cursorgröße in Rasterzeilen
	0062	0062	Nummer der aktuellen Bildschirmseite
	00F0	00FF	für Benutzer reseviert
	Hauptspeicher		
4000	FFFF		Ende des RAM-Bereichs bei 256 KByte
8000	FFFF		Ende des RAM-Bereichs bei 512 KByte
9000	FFFF		Ende des RAM-Bereichs bei 640 KByte
	Video- und ROM-Bereich		
A000	0000	FFFF	Erweiterungsbereich für Grafik-Adapter
B000	0000	0FFF	Bildschirmspeicher Monochrom-Adapter
B800	0000	3FFF	Bildschirmspeicher Farbgrafik-Adapter
C800	0000	7FFF	Erweiterungs-ROM (Festplattencontroller etc.)
D000	0000	FFFF	Extended-Memory-Bereich 1
E000	0000	FFFF	Extended-Memory-Bereich 2
FC00	0000	3FFF	BIOS-ROM

Tabelle 1. Interessante Systemadressen beim PC

Kommando	Abkürzung	Funktion
Break	-	Schaltet die Wirkung von [Ctrl-C] ein oder aus
Chdir	cd	Wechselt oder zeigt das aktuelle Verzeichnis
Cls	-	Löscht den Bildschirm
Copy	-	Kopiert Dateien
Ctty	-	Leitet Bildschirmausgabe um
Date	-	Setzt oder zeigt das Systemdatum
Dir	-	Zeigt den Inhalt eines Verzeichnisses
Erase	del	Löscht Dateien
Exit	-	Beendet »command.com«
Mkdir	md	Erzeugt ein neues Verzeichnis
Path	-	Legt den Suchpfad fest
Prompt	-	Legt den MS-DOS-Promptstring fest
Rename	ren	Benennt Dateien um
Rmdir	rd	Löscht ein (leeres) Verzeichnis
Set	-	Zeigt oder setzt Umgebungsvariablen
Time	-	Zeigt oder setzt die Systemuhrzeit
Type	-	Zeigt Textdateien
Ver	-	Zeigt die MS-DOS-Versionnummer
Vol	-	Zeigt die Kennung der Diskette oder Festplatte

Tabelle 2. Die residenten DOS-Kommandos

zwischen den einzelnen Segmenten gibt. Doch auch Überlappungen sind möglich: Ein 64-KByte-Segment kann an jeder beliebigen durch 16 teilbaren Adresse im 1-MByte-Speicherbereich beginnen. Eine absolute Speicheradresse setzt sich daher aus der Segmentbasisadresse und der relativen Entfernung zur gewünschten effektiven Adresse, dem Offset, zusam-

men (Bild 2). Die absolute Adresse läßt sich also nach folgender Formel errechnen:

Absolute Adresse =
Segmentbasisadresse x 16 + Offset

Sowohl die Segmentbasisadresse als auch der Offset sind 16-Bit-Werte und können vom 8088/8086 bequem gehandhabt werden. Jede einzelne Adresse innerhalb eines Speichers

von 1 MByte läßt sich so erreichen. Durch die Multiplikation mit 16, die die Adreßlogik des Prozessors automatisch bei jedem Speicherzugriff durchführt, wird aus der 16-Bit-Segmentadresse eine 20-Bit-Adresse, deren vier letzte Bit immer Null sind. Die Multiplikation mit 16 entspricht nämlich im binären Zahlensystem gerade einer Verschiebung um vier Bit-närstellen nach links.

Speicheradressen werden wegen dieser Aufspaltung in Segment- und Offset-Adresse bei einem 8088/8086-System meist in zusammengesetzter Form angegeben. Die effektive Speicheradresse 789AB (hexadezimal) beispielsweise wird dargestellt als »789A:000B«. Die Segmentadresse ist hier 789A, der Offset innerhalb dieses Segmentes ist 000B oder 11 dezimal. Die Adresse bezeichnet also das elfte Byte im 64-KByte-Segment, das bei Adresse 789A0 beginnt. Die Darstellung durch Segment- und Offset-Adresse ist nicht eindeutig. Auch »7890:00AB« oder »7000:89AB« bezeichnen die gleiche Adresse. Sie sehen, daß die Speicherverwaltung bei diesem Prozessor sehr kompliziert werden kann.

Auf Assembler-Ebene werden fast alle Adressen als 16-Bit-Daten behandelt. Diese Daten stellen dann den Offset-Teil der Adresse dar. Die Segmentbasisadresse dazu wird aus einem von insgesamt vier speziellen Segment-Registern entnommen. Jedes dieser vier Segment-Register ist auf eine spezielle Ausgabe zugeschnitten. Bild 3 verdeutlicht die Verhältnisse.

Vier Segmentregister organisieren beim 8088/8086 alle Speicherzugriffe

Das CS-Register bestimmt das Codesegment, also das 64-KByte-Segment, in dem sich das gerade ablaufende Maschinenprogramm befindet. Mit dem CS-Register korrespondiert das IP-Register (Instruction-Pointer). IP stellt den Offset-Anteil des Befehlszeigers dar. Der jeweils zu bearbeitende Maschinenbefehl wird also aus der effektiven Adresse CS:IP geholt. Alle Sprungbefehle und Unterprogrammaufrufe beziehen sich normalerweise auf das Codesegment.

Das DS-Register dagegen bestimmt, aus welchem Speichersegment die Daten geholt werden, mit denen das Programm arbeiten soll. Alle Maschinenbefehle, die Daten laden oder verändern, beziehen sich auf das Datensegment. Mit dem DS-Register korrespondieren die BX-, SI- und DI-Register, wenn diese zum Adressieren benutzt werden.

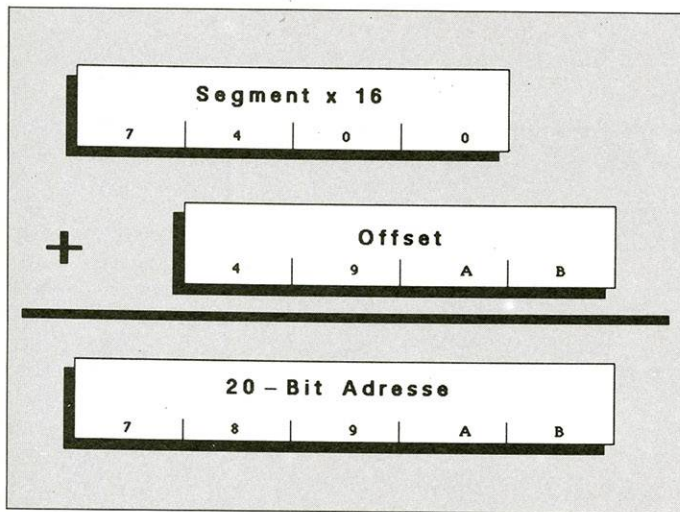


Bild 2.
Zusammensetzung
der effektiven
20-Bit-Adresse

Das SS-Register zeigt auf das Stapel-segment. Rückkehradressen von Unterprogrammen und zwischengespeicherte Daten werden in diesem Segment abgelegt. Mit SS ist das SP-Register (Stack-Pointer) verbunden, das die Offset-Adresse des Stapelzeigers enthält. Die effektive Adresse des Prozessorstapels im Speicher ist also SS:SP. Das BP-Register wird auch zusammen mit dem SS-Register als Adresse verwendet; dieses Paar wird häufig eingesetzt, um Werte aus dem Stack zu lesen.

Das vierte Segment-Register schließlich ist das ES-Register, das das sogenannte Extra-Segment bestimmt. Dieses Extra-Segment hat keinen speziellen Aufgabenbereich, sondern wird vom Programmierer je nach aktuellen Erfordernissen eingesetzt. Normalerweise dient es als Ergänzung des DS-Registers, um den Datenspeicherbereich über 64 KByte hinaus zu erweitern.

Die Programmierung des 8088/8086 wird sehr flexibel durch die Tatsache, daß jeder Befehl bei Bedarf zwangsweise auf ein bestimmtes Segment-Register wirken kann. Beispielsweise kann der Programmierer angeben, daß ein MOV-Befehl, der Daten in ein Register lädt, diese Daten aus dem Codesegment holt statt aus dem Datensegment, wie es der Normalfall wäre. Dies geschieht in Assemblersprache durch den sogenannten »Segment-Override-Prefix«.

Die Segment-Architektur der Intel-Prozessoren macht sich auch beim Programmieren in Hochsprachen bemerkbar

Auch bei der Programmierung in einer Hochsprache macht sich die Segment-Architektur des 8088/8086 in vielerlei Hinsicht deutlich bemerk-

bar. Turbo Pascal 3.0 beispielsweise erlaubt eine maximale Programmgröße von 64 KByte, weil das Codesegment durch Turbo Pascal 3.0 nicht manipuliert wird. Dies bringt kompakten und schnellen Programmcode, aber eben auch eine Begrenzung der möglichen Programmgröße. Aus der Speichersegmentierung resultiert auch die bei fast allen Programmiersprachen für den PC vorhandene Beschränkung auf Datenstrukturen kleiner als 64 KByte. Beispielsweise ist unter Turbo Pascal ein »array[1..70000] of char« nicht erlaubt. Turbo Pascal erlaubt allerdings sehr bequem den direkten Zugriff auf

Speicherstellen durch das MEM-Array. Die Zuweisung

```
»a := mem[$B800:$0000]«
```

liest beispielsweise ein Byte aus der Adresse »B800:0000«, dem Beginn des Bildschirmspeichers bei der Farbgrafikkarte. Auch Pointer sind unter Turbo Pascal ähnlich organisiert und bestehen aus einem Segment- und einem Offset-Teil.

Etwas anders sieht es bei Basic aus. Basic und GW Basic behandeln sowohl das Basic-Programm als auch die Variablen als Daten und legen sie infolgedessen einfach im Datensegment ab. Das Codesegment CS zeigt auf den Basic-Interpreter selbst. Dies hat zur Folge, daß für Programm und Daten zusammen nur 64 KByte Speicher zur Verfügung stehen. Effektiv ist es sogar noch weniger, denn der Basic-Interpreter reserviert noch etwas vom Datensegment für seine internen Operationen. Um direkt auf den Speicher zuzugreifen, bietet Basic die Operationen »peek« und »poke«, die beide auf das durch die »def seg«-Anweisung definierte Speichersegment wirken.

Bei Quick Basic und Turbo Basic ist die Programmgröße ebenso wie bei Turbo Pascal 4.0 und den meisten anderen modernen Programmiersprachen nicht mehr auf 64 KByte begrenzt, da diese Sprache alle Exe-Da-

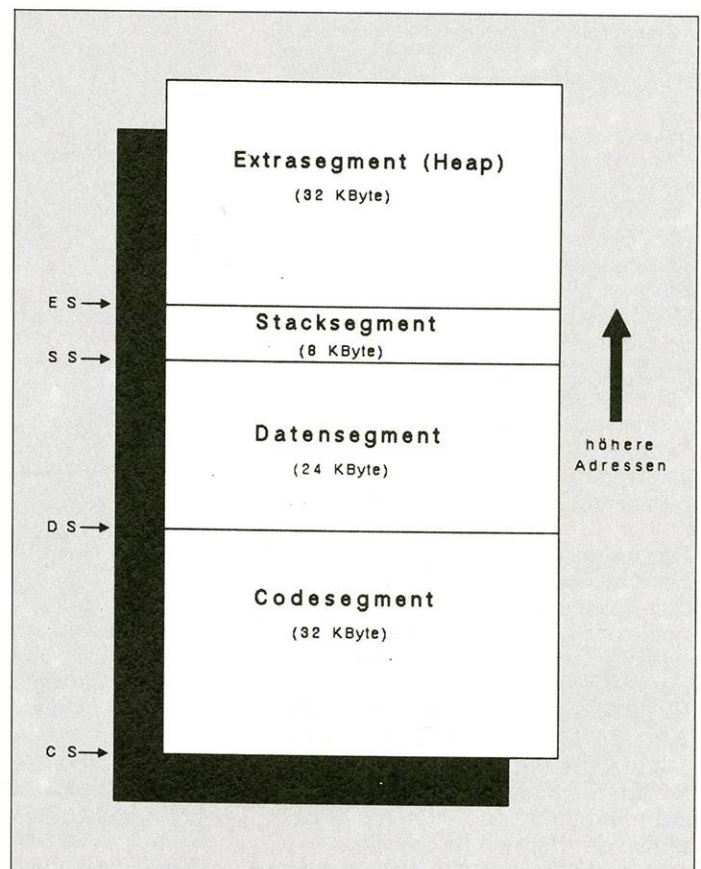


Bild 3.
Eine typische
Segment-
aufteilung.

teilen als ausführbare Programme erzeugen. Ein solches Exe-Programm kann, etwas vereinfacht gesprochen, aus mehreren Modulen bestehen, die jedes für sich ein Codesegment mit maximal 64 KByte Umfang belegen können.

Eine Wanderung durch den Hauptspeicher des PC führt an vielen interessanten Adressen vorbei

Mit einem MByte Speicher läßt sich eine ganze Menge anfangen. Immerhin ist dies das 16fache dessen, was ein 8-Bit-Computer ohne aufwendige Verrenkungen ansprechen kann. Bild 4 gibt Ihnen einen ersten Überblick über die Belegung dieses großen Adressenbereiches. Im Rahmen dieses Sonderheftes wird der eine oder andere Speicherbereich noch intensiver besprochen. An dieser Stelle jedoch erhalten Sie zunächst einige grundsätzliche Informationen über die Speicheraufteilung Ihres PC.

Der gesamte Speicher wird üblicherweise in 16 Blöcke zu je 64 KByte unterteilt dargestellt. Die Benennung der einzelnen Speicherblöcke erfolgt dabei einfach nach der ersten (hexadezimalen) Ziffer der Basisadresse des jeweiligen Blocks. Der erste Speicherblock wird also 0-Block genannt, der letzte F-Block. Zum F-Block gehören alle Adressen von F0000 bis FFFFF oder in Segment-Schreibweise: »F000:0000« bis »F000:FFFF«.

Die ersten zehn Speicherblöcke, also der Adreßbereich von 00000 bis 9FFFF, sind bei allen IBM-kompatiblen PCs für Schreib-/Lesespeicher (RAM) reserviert. Daraus ergibt sich jene magische Grenze von 640 KByte (10 x 64 KByte) für den maximalen RAM-Ausbau. Die unteren Adressen im 0-Block sind dabei als Arbeitsbereich für das System reserviert. Tabelle 1 zeigt einige interessante Systemadressen; im Laufe dieses Kurses werden Sie noch mehr Systemadressen und deren Anwendungen kennenlernen.

Die insgesamt 128 KByte des A- und B-Bereichs dienen Zwecken der Video-Ansteuerung. Der einfache Monochrom-Adapter (MDA) beispielsweise belegt 4 KByte im Adreßbereich ab B0000, während der Farbgrafik-Adapter (CGA) 16 KByte ab der Adresse B8000 verwendet. Die Hercules-Karte (MGA) belegt gleich 64 KByte, nämlich den gesamten Adreßbereich von B0000 bis BFFFF, wobei der Bereich in zwei Bildschirmseiten zu je 32 KByte aufgeteilt

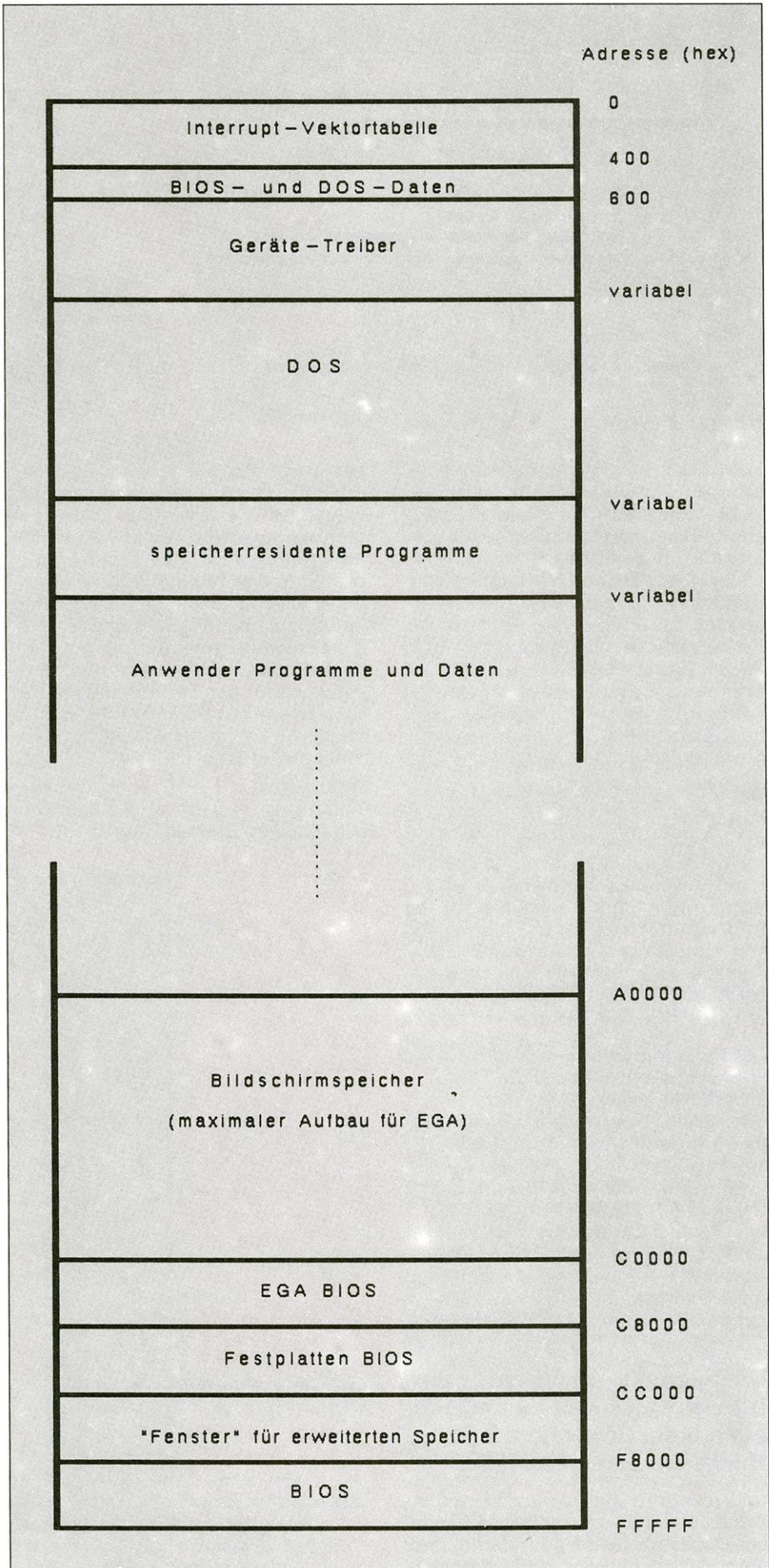


Bild 4. Die Aufteilung des PC-Speichers

Offset hex	dez	Bytes	Inhalt
00	0	2	INT-20hex-Anweisung
02	2	2	freier Speicher (in Paragraphs)
04	4	1	(reserviert)
05	5	5	MS-DOS-Aufruf (Far Call)
0A	10	4	Ende-Vektor
0E	14	4	Unterbrechungs-Vektor
12	18	4	Fehler-Vektor
16	22	22	Arbeitsbereich MS-DOS
2C	44	2	Segment-Adresse Umgebungsstring
2E	46	34	Arbeitsbereich für MS-DOS
50	80	3	INT-33hex-Anweisung mit Return
53	83	2	(reserviert)
55	85	7	Erweiterung File Control Block 1
5C	92	9	File Control Block 1 (FCB 1)
65	101	7	Erweiterung File Control Block 2
6C	108	20	File Control Block 2 (FCB 2)
80	128	1	Parameterlänge
81	129	127	Parameterstring
80	128	128	Diskettentransferbereich (DTA)

Tabelle 3. Der Aufbau des Program Segment Prefix (PSP)

Register COM-Datei	Belegung EXE-Datei
CS	PSP (Linker)
DS	PSP
SS	PSP (Linker)
ES	PSP
IP	0100hex (Linker)
SP	FFFEhex (Linker)

Tabelle 4. Die Registerbelegung beim Programmaufruf

Offset hex	dez	Bytes	Inhalt
00	0	2	EXE-Kennung (»MZ«)
02	2	2	Anzahl Bytes in letzter Page
04	4	2	Anzahl der 512-Byte-Pages
06	6	2	Anzahl der Relokations-Einträge
08	8	2	Header-Größe in Paragraphs
0A	10	2	minimaler Speicherbedarf in Paragraphs
0C	12	2	maximaler Speicherbedarf in Paragraphs
0E	14	2	Startwert für SS-Register
10	16	2	Startwert für SP-Register
12	18	2	negative Prüfsumme über alle Worte
14	20	2	Startwert für IP-Register
16	22	2	Startwert für CS-Register
18	24	2	Offset der Relokationstabelle
1A	26	2	Overlay-Nummer

Tabelle 5. Der Aufbau eines EXE-Headers

ist. Durch diesen großen Grafik-Speicherbereich ist die Grafikauflösung der Hercules-Karte deutlich besser als bei der Farbgrafikkarte.

Die in neuerer Zeit ebenfalls häufig zu findende EGA-Karte ist unter anderem in der Lage, die Farbgrafikkarte vollständig zu emulieren. EGA-Karten stellen je nach Ausbaustufe bis zu 256 KByte Grafikspeicher zur Verfügung, der jedoch nicht einfach in den Adreßbereich des PCs eingebunden ist, sondern über komplizierte Verfahren des »Memory Mapping«

verwaltet wird. Grundsätzlich ist der gesamte A-Bereich des Prozessors für spezielle Grafik-Erweiterungen vorgesehen.

Der C-Speicherblock ist beim einfachen PC nicht belegt. Er ist für spezielle Betriebssystem-Erweiterungen vorgesehen. Beispielsweise ist in diesem Bereich die Software für die XT-Festplattenstation untergebracht. Die Speicherblöcke D und E sind beim IBM-PC für steckbare ROM-Module reserviert. Außer beim IBM-PC Junior, dem ja wenig Erfolg beschieden

Software Tools Pascal, Turbo Pascal

- ▶ **TDebug PLUS 4.0** ab DM 115,-
Symbolischer Debugger
- ▶ **Turbo Professional 4.0** DM 299,-
300 Utilities für Turbo Pascal 4.0
- ▶ **Overlay Manager** DM 115,-
für Turbo Pascal 4.0
- ▶ **BTree Isam** DM 444,-
Dateiverwaltungs-System für Turbo Pascal 3.0/4.0, Modula
- ▶ **Netzmodul zur BTree Isam** DM 555,-
für beliebig viele Workstations
- ▶ **Turbo MAGIC** DM 295,-
Masken und Menue-Generator mit deutschem Zeichensatz für Turbo Pascal 3.0/4.0, Modula

C, Turbo C

Turbo Zorlite Lattice Microsoft **C** + Tools
zum Beispiel

- ▶ **C Asynch Manager** (Blaise) DM 395,-
Serielle Datenkommunikation für Turbo C und C
- ▶ **C Tools Plus** (Blaise) DM 295,-
200 Routinen für Turbo C, C
- ▶ **BTree Isam** DM 444,-
Bewährt unter Turbo Pascal. Jetzt auch für C mit Source
- ▶ **Netzmodul zur BTree Isam** DM 555,-
für beliebig viele Workstations
- ▶ **Pascal-C** DM 498,-
Turbo Pascal 3.0/4.0 -> C

Aktuell

NEU

TURBO
Power

Turbo Analyst

7 unentbehrliche Utilities für Turbo Pascal 4

- ▶ **PSA-** Deckt subtile Codierprobleme auf
- ▶ **PF-** Formatiert Sourcecode
- ▶ **PLIST-** Formatiert die Druckerausgabe von Sourcecode
- ▶ **PEP-** Ermittelt das Laufzeitverhalten von Programmen
- ▶ **MON-** Ermittelt die Aufrufhäufigkeit von Anweisungen
- ▶ **EXE-** Gibt detaillierte Information über EXE-Files
- ▶ **INFO-** Gibt detaillierte Information über TPU-Files
- + **INTEGRATOR-** Integriert alle Utilities in einer Benutzeroberfläche

Deutsche Version incl. Sourcecode
Einführungspreis: DM 195,-

Fordern Sie kostenlose Informationen an

ENZ
EDV-BERATUNG GMBH

6380 Bad Homburg 6
Wetterauer Str. 12
☎ 061 72 / 4 64 85

war, wird diese Option jedoch praktisch nicht verwendet. Dafür wird dieser Bereich durch den Speichererweiterungs-Standard EMS benutzt. Dieser Standard arbeitet vereinfacht gesagt so, daß ein sehr großer, möglicherweise mehrere MByte umfassender Speicherbereich durch eine spezielle Hard- und Software so verwaltet wird, daß die gerade vom Prozessor benötigten Segmente des erweiterten Speichers in den Hauptspeicherbereich zwischen D0000 und EFFFF einblendet werden.

Im letzten Speicherblock ist in einem gewissen Sinne das Herz des PC untergebracht. Der F-Bereich enthält das BIOS-ROM, welches die ganz grundlegenden Funktionen des Computers steuert. Beispielsweise sind hier die Routinen vorhanden, um die Tastatur abzufragen, um Zeichen auf den Bildschirm auszugeben oder Diskettensektoren zu lesen und zu schreiben.

Damit Ihr PC sinnvolle Arbeit leisten kann, beispielsweise Befehle von Ihnen entgegennimmt oder komplexe Programme ausführt, ist ein Betriebssystem erforderlich. Bekannte Betriebssysteme für den PC sind MS-DOS, PC-DOS, CP/M 86 und – in neuester Zeit – OS/2 oder BS/2. Von diesen Systemen sind MS-DOS und PC-DOS am weitesten verbreitet, da jeder PC mit einem dieser beiden Systeme ausgeliefert wird.

MS-DOS und PC-DOS sind weitgehend identisch aufgebaut. Der grundlegende Unterschied ist eigentlich nur der, daß PC-DOS das Betriebssystem für die IBM-PCs ist, während MS-DOS mit IBM-kompatiblen Computern ausgeliefert wird. Beide Betriebssysteme werden wir im Laufe des Kurses aus Vereinfachungsgründen einfach als MS-DOS bezeichnen, was IBM uns nachsehen möge. Das Betriebssystem der PCs besteht aus mehreren Teilen:

- dem BIOS (Basic Input/Output-System) im ROM
- dem eigentlichen DOS (Disk Operating System)
- dem Kommandoprozessor »command.com«

Beginnen wir bei der weiteren Betrachtung des Betriebssystems mit dem Kommandoprozessor. Wie die Namenserverweiterung COM bereits andeutet, handelt es sich dabei um ein ganz normales MS-DOS-Programm. Tatsächlich benutzt Command die Funktionen des MS-DOS-Systems, um den Kontakt mit dem Computerbenutzer herzustellen. Erst Command erlaubt Ihnen die Kommunikation

mit dem Computer, ihm Befehle zu geben und Programme zu starten.

Durch den Kommandoprozessor »command.com« kann der Benutzer mit dem PC kommunizieren und Leistungen des Betriebssystems anfordern

Wenn Sie eine Systemdiskette (Boot-Diskette) herstellen, entweder durch Formatieren mit der »/S«-Option oder mit Hilfe der Sys-Anweisung, werden nicht nur zwei versteckte Dateien angelegt, die das Betriebssystem beinhalten, sondern es wird auch die Datei »command.com« auf die neue Boot-Diskette übertragen.

Die beiden versteckten Dateien enthalten das eigentliche Betriebssystem, das zu Anfang wie ein normales Programm in den Computer geladen wird. Beim Starten des Computers wird, nachdem das Betriebssystem geladen ist, sofort ein weiteres Programm aufgerufen. Dabei handelt es sich um das bereits zitierte Command. Dieses Programm ist es auch, daß Ihnen den Prompt (»>«) am Bildschirm präsentiert. Doch nur dafür würde kein Programm 20 KByte und mehr (je nach Version) Speicherplatz benötigen.

Command beinhaltet vielmehr auch noch die residenten MS-DOS-Kommandos. Dabei handelt es sich um Programme, die nicht erst von der Diskette oder Festplatte zu laden sind, sondern die unmittelbar ausgeführt werden können. Diese bekannten Kommandos sind in Tabelle 2 aufgeführt. Außerdem dient Command auch als Kommandointerpreter. Das bedeutet, es untersucht alle Ihre Eingaben nach Sinn und Zweck und führt die darin enthaltenen Befehle aus.

Dabei ist eine Menge zu tun. Es muß entschieden werden, ob es sich um ein residentes oder ein transientes (also ein erst zu ladendes) Programm handelt. Des weiteren müssen eventuell angegebene Parameter an das aufgerufene Programm übergeben werden. Und das wichtigste ist natürlich das Laden und Starten des gewünschten Programmes oder die Ausgabe einer entsprechenden Fehlermeldung.

Sie sehen also, daß die Sache mit dem Aufruf von Programmen und Befehlen gar nicht so einfach abläuft. Sie sehen aber auch, daß MS-DOS einige sehr wichtige und eifrige Helfer zur Seite hat, die alle Operationen optimal unterstützen. Command ist aber nicht nur ein Programm, das ab und zu nachgeladen wird, sondern kann auch vom Anwender selbst eingesetzt werden. Sie

können also auch Command tatsächlich wie jedes andere MS-DOS-Kommando aufrufen. Stellen Sie sich Command einfach als spezielles Kommando vor, das die Eigenschaft besitzt, andere MS-DOS-Befehle aufrufen zu können. Versuchen Sie es selbst einmal, indem Sie einfach das Kommando »command« eingeben. Es erscheint nach kurzer Zeit die Einschaltmeldung des Kommandoprozessors und das gewohnte Prompt. Falls Sie dagegen eine Meldung über ein ungültiges Kommando erhalten, dann überprüfen Sie, ob Sie auch eine Diskette mit »command.com« eingelegt haben, und ob Sie sich im richtigen Unterverzeichnis (Subdirectory) befinden.

Beim Aufruf von Command stehen mehrere Parameter zur Verfügung. Sobald Sie Command ein zweites Mal aufrufen, wird automatisch eine neue Ebene geschaffen, die mit dem residenten Befehl »Exit« wieder verlassen werden kann. Jede solche Ebene belegt Speicherplatz in der Größenordnung der Command-Datei, der erst beim Exit-Kommando wieder freigegeben wird. Was bringt nun das Eröffnen einer neuen Ebene für den Benutzer?

Neben den Aufgaben, die bei der Verwaltung und Ausführung der residenten Befehle durchzuführen sind, merkt sich Command auch noch den verwendeten Gerätetreiber (Steuerprogramme für die Tastatur, den Bildschirm, das Diskettenlaufwerk und so weiter), sowie die Größe des sogenannten Environment-Bereichs. Der Environment-Bereich (Umgebungs-Bereich) enthält Angaben zur aktuellen MS-DOS-Arbeitsumgebung, beispielsweise von welchem Laufwerk/Directory bei Bedarf »command.com« nachgeladen werden kann. Daneben merkt sich MS-DOS hier die mit »path« definierten Zugriffspfade innerhalb der einzelnen Verzeichnisse (Directories) und alle mit »set« explizit gesetzten Umgebungs-Variablen. Durch Eingabe von »set« ohne weitere Parameter gibt MS-DOS Ihnen eine Auflistung des Inhalts der Arbeitsumgebung.

Mit Basic kann der Environment-Bereich einfach und zielsicher in Programmen manipuliert werden

GW-Basic und Basica sind in der Lage, direkt auf diesen Umgebungsbereich zuzugreifen. Dazu dient die Basic-Funktion »environ\$« und der Befehl »environ«. Mit dem Environ-Befehl wird eine Stringvariable in den Umgebungsbereich mit aufgenommen. Ein Löschen ist jedoch

nicht vorgesehen; deshalb ist dieser Befehl nur mit Vorsicht zu verwenden. Listing 1 verdeutlicht die Anwendung der Environ\$-Funktion, mit der sich die MS-DOS-Umgebung lesen läßt.

Mit den meisten MS-DOS-Versionen läßt sich die Größe des Environment-Bereichs beim Aufruf von Command über einen bestimmten Parameter festlegen. Je größer dieser Bereich, desto länger können zum Beispiel die Pfade sein, die Sie eventuell beim Starten des Systems mittels Path-Kommando einrichten. Die Größe des Environment-Bereichs ist standardmäßig auf 160 Byte eingestellt. Ab MS-DOS-Version 3.0 kann dieser Wert durch die Option »e« beim Aufruf von Command auf bis zu maximal 32768 Byte (MS-DOS Versionen 3.2 und 3.3) beziehungsweise 992 Byte (MS-DOS-Versionen 3.0 und 3.1) erhöht werden. Das folgende Kommando startet den Kommandoprozessor neu und legt einen erweiterten Environment-Bereich von 512 Byte fest:

```
command /e:512
```

(Versionen 3.2 und 3.3) oder mit den MS-DOS-Versionen 3.0 und 3.1:

```
command /e:32
```

Das Kommando für die neueren Versionen 3.2 und 3.3 wird also mit der Länge in Bytes angegeben und ist ein dokumentierter Bestandteil von MS-DOS. Anders sieht es bei den älteren Versionen aus: Die Länge ist in Paragraphen (16-Byte-Einheiten) anzugeben, und das Kommando findet nirgends in der MS-DOS-Dokumentation irgendeine Erwähnung.

Doch nun zu weiteren interessanten Optionen. Starten Sie doch einmal »command.com« von neuem und verwenden Sie dabei den Parameter »p«:

```
command /p
```

Es wird eine völlig neue Kopie des Kommandoprozessors geladen und die alte Ebene wird vollständig gelöscht. Sie können also in diesem Fall zum Beispiel nicht mehr auf die zuvor definierten Zugriffspfade zurückgreifen. Auch ein Verlassen des Kommandoprozessors mittels des Exit-Kommandos ist nicht mehr durchführbar.

Der letzte mögliche Parameter schließlich erlaubt es, eine neue Ebene zu öffnen, dort einen MS-DOS-Befehl auszuführen und anschließend sofort auf die aktuelle Ebene zurückzukehren. Probieren Sie einmal folgendes aus:

```
command /c dir
```

Mit diesem Befehl laden Sie eine neue Kopie des Kommandoprozessors, von der der Befehl »dir« ausgeführt wird. Danach kehrt das System automatisch wieder auf die vorherige Ebene zurück.

Natürlich gibt es unter MS-DOS einfachere Wege, um an das Inhaltsverzeichnis einer Diskette zu kommen. Die Fähigkeit von MS-DOS, jederzeit den Kommandoprozessor aufrufen zu können, ist auch nicht in erster Linie für die Anwendung durch direkte Kommandos gedacht. Vielmehr gibt es hierdurch ein einfaches, aber wirkungsvolles Konzept, um alle MS-DOS-Befehle auch aus Anwenderprogrammen heraus aufrufen zu können. Viele Programme machen davon auch Gebrauch und erlauben dem Anwender den Aufruf von Betriebssystem-Funktionen. Dies geschieht vom Programm aus sehr einfach: Das Programm ruft nur den Kommandoprozessor in der vorhin beschriebenen Art und Weise auf. Der Benutzer kann jetzt alle MS-DOS-Befehle nutzen, sich Inhaltsverzeichnisse ansehen und Dateien hin und her kopieren. Sobald er damit fertig ist, gibt er den Befehl »exit« ein, und die zuvor eingerichtete Command-Ebene wird wieder verlassen. Der Benutzer findet sich ohne Umschweife im laufenden Anwenderprogramm wieder.

Unter Basic kann man sich Command über das Shell-Kommando zunutze machen. »shell« lädt und startet den Kommandoprozessor im Direktmodus oder im laufenden Programm. Als Parameter kann jeder MS-DOS-Befehl als String angegeben werden. Bei Aufruf ohne Parameter wird Command aufgerufen, und Sie können nun ganz normal MS-DOS-Befehle eingeben. Durch den Exit-Befehl gelangen Sie wieder in das laufende Programm zurück. Listing 2 zeigt ein Beispiel für die Anwendung des Shell-Kommandos. Turbo Pascal 3.0 bietet leider standardmäßig keine entsprechende Fähigkeit, jedoch kann mit einem kleinen Programm Command natürlich auch von Turbo Pascal aus genutzt werden. Listing 3 stellt Ihnen diese Funktion für Ihre eigenen Programme zur Verfügung.

Der Kommandoprozessor behandelt Bat-, Com- und Exe-Dateien jeweils völlig unterschiedlich

Wie bereits erwähnt, beinhaltet Command die Schnittstelle des Betriebssystems zum Anwender. Alle Ihre Eingaben werden von diesem Programm interpretiert. Was passiert nun, wenn Command feststellt, daß ein Programm von der Diskette geladen werden muß? Prinzipiell gibt es drei Arten von ausführbaren Programmen, die anhand der Namensweiterung (Extension) unterschieden werden. Es sind dies die Com-, Exe- und Bat-Dateien. Alle drei Dateitypen

werden von Command unterschiedlich behandelt. Sobald Command feststellt, daß ein vom Anwender eingegebener Befehl nicht zu den residenten Kommandos gehört, wird versucht, eine Com-Datei dieses Namens zu finden. Wenn dieser Versuch fehlschlägt, wird es mit einer Exe-Datei versucht. Klappt auch das nicht, dann sucht Command schließlich nach einer Bat-Datei des angegebenen Namens und gibt bei einem neuerlichen Fehlschlag eine Fehlermeldung aus.

Am einfachsten ist der Ablauf im Fall von Bat-Dateien. Solche Dateien enthalten MS-DOS-Kommandos im Klartext, die einfach nacheinander abgearbeitet werden. Wir wollen uns an dieser Stelle aber nicht weiter mit Bat-Dateien beschäftigen, sondern wenden uns gleich den Com- und Exe-Dateien zu.

Lädt Command ein Programm mit der Namensweiterung »com«, so werden alle vier Segment-Register auf die erste freie Adresse im Speicher gesetzt. Da die ersten 256 Byte in jeder Com-Datei für den sogenannten »Program Segment Prefix« (Tabelle 3) reserviert sind, wird der Programmzähler auf einen Offset von 100hex (entspricht dezimal 256) eingestellt. Der Stapelzeiger wird auf das Ende des aktuellen 64-KByte-Segments gesetzt. Das erste (und zu diesem Zeitpunkt das einzige) Wort im Stapel ist 0. Ein Assembler-Programm, das den Stapel nicht verändert, springt also über ein einfaches Return zur Adresse 0 im Codesegment. Das ist aber gerade der Anfang des PSP, wo ein Befehl »INT 20h« steht. Dieser Interrupt bewirkt die Rückkehr zum Betriebssystem. Diese Methode des Beendens eines Programms wird aber nicht mehr empfohlen. Statt dessen sollte die entsprechende MS-DOS-Funktion (Nummer 0) dafür verwendet werden. In Listing 4 sehen Sie ein Programm in Turbo Pascal, das den Program Segment Prefix von Turbo ausliest und anzeigt.

Die Belegung der wichtigsten Prozessor-Register beim Start von Com- und Exe-Programmen ist in Tabelle 4 angegeben. Bei einem Com-Programm wird im übrigen der gesamte zur Verfügung stehende Speicherplatz Ihres Computers reserviert. Es wird auch der vorher von Command belegte Speicherplatz verwendet. Wenn das aufgerufene Programm nicht von sich aus einen Teil des Speichers wieder an MS-DOS zurückgibt, dann kann es vorkommen, daß nach der Ausführung Ihres COM-Programmes Command neu geladen werden muß, damit die residenten Befehle von MS-DOS wieder zur Verfügung stehen.

Noch etwas kann sehr schnell zur Verwirrung des Programmierers beitragen. MS-DOS bietet dem Assembler-Programmierer verschiedene Wege, sein Programm zu beenden. Einer davon ermöglicht es, das Programm weiterhin im Speicher zu behalten. Wird daraufhin ein neues Programm geladen und ist nicht mehr genügend Speicherplatz vorhanden, stürzt das sonst so zuverlässige Betriebssystem total ab. Hier hilft nur noch ein Rücksetzen des Computers mit den drei magischen Tasten [Ctrl-Alt-Del].

Etwas anders verhalten sich Programme, die mit der Extension »exe« im Directory auftauchen. Am Anfang stehen hier immer ein Datei-Kopf

(Header), dem das eigentliche Programm als Lade-Modul folgt. Im Header befinden sich Informationen über die Anzahl der Bytes in der letzten Speicherseite des Programms sowie die Länge der Datei in Speicherseiten zu je 512 Byte. Tabelle 5 zeigt den Aufbau des Exe-Headers.

Listing 5 enthält ein Basic-Programm zum Lesen und Ausgeben der Header-Informationen von Exe-Dateien. Wenn Sie mit diesem Listing etwas experimentieren, dann werden Sie feststellen, daß eine ganze Reihe von Exe-Programmen, insbesondere die kürzeren, nur wenig Gebrauch von den einzelnen Feldern des Exe-Headers machen.

Gerade kurze Utilities sind häufig schon voll relocatibel geschrieben und müssen nicht über den Exe-Lader bearbeitet werden. In solchen Fällen bleiben die entsprechenden Felder des Exe-Headers unbesetzt. Hinter dem Header folgt dann das Lade-Modul, welches das eigentliche Programm in relocatibler, also frei im Speicher verschiebbarer Form enthält. Exe-Programme sind nicht an die Begrenzung auf einen 64-KByte-Block für Programm oder Daten gebunden. Dadurch sind sie in der Regel um einiges länger und auch etwas langsamer als entsprechende Com-Programme.

(Michael Seyler/
Thomas Little/ev)

```
[3T4] 10 REM Environ-Demo
[1B1] 20 REM
[E4G] 30 PRINT:PRINT"Auflistung der DOS-Umgebung von Gw
basic"
[DF2] 40 PRINT:I=1
[K06] 50 WHILE LEN(ENVIRON$(I)) > 0
[356] 60 PRINT I,ENVIRON$(I):I=I+1
[WU1] 70 WEND
[SLE] 80 PRINT:PRINT"Keine (weiteren) Umgebungsvariable
n"
[BX1] 90 PRINT
```

Listing 1. Anwendung der Environ\$-Funktion von Basic

```
[YPB] 10 REM Demonstration des Shell-Befehls
[1B1] 20 REM
[OU1] 30 KEY OFF
[CF6] 40 INPUT"DOS-Kommando: ",B$
[K25] 50 IF B$="end" THEN END
[WD3] 60 IF B$="" THEN 40
[HNA] 70 PRINT"Kommando wird ausgeführt..."
[YU5] 80 SHELL B$:PRINT:PRINT
[N22] 90 GOTO 40
```

Listing 2. Ein Beispiel für das Shell-Kommando

```
[AM3] 0: program PSP;
[000] 1:
[V1D] 2: { Dieses Programm zeigt seinen eigenen }
[GLA] 3: { Program-Segment-Prefix (PSP) an }
[757] 4: { Volker Everts, 4.4.87 }
[000] 5:
[A21] 6: type
[HP5] 7: HexStr = string[4];
[W77] 8: ZeigerStr = string[99];
[7N4] 9: FarAdr = ^Byte;
[000] 10:
[TE5] 11: FarCall = record
[3V3] 12: Befehl: byte;
[XV2] 13: Adr: FarAdr;
[TU0] 14: end;
[000] 15:
[S75] 16: PSPtyp = record
[S74] 17: int20: integer;
[CT5] 18: Segment: integer;
[NN5] 19: reserviert: byte;
[E63] 20: DOS: FarCall;
[W53] 21: Ende: FarAdr;
[E63] 22: Abbr: FarAdr;
[KY4] 23: Fehler: FarAdr;
[LM8] 24: intern: array[1..20] of byte;
[AV5] 25: Umgebung: integer;
[TU0] 26: end;
[000] 27:
[000] 28:
[EM0] 29: var
[L97] 30: PSP: PSPtyp absolute CSeg:0;
[M3D] 31: { Der PSP liegt am Anfang des Codesegment
s }
[000] 32:
[000] 33:
[000] 34:
[1VA] 35: function hex(x: integer): HexStr;
```

```
[000] 36:
[V5D] 37: { wandelt x in hexadezimale Darstellung }
[000] 38:
[VM1] 39: const
[1NA] 40: HexZiffer: string[16] = '0123456789ABCDEF
';
[000] 41:
[EM0] 42: var
[W33] 43: i: integer;
[KK2] 44: h: HexStr;
[000] 45:
[BH1] 46: begin
[W16] 47: for i := 4 downto 1 do
[BH1] 48: begin
[EP8] 49: h[i] := HexZiffer[x and 15 + 1];
[DK2] 50: x := x shr 4;
[TU0] 51: end;
[S02] 52: h[0] := #4;
[SPl] 53: hex := h;
[TU0] 54: end;
[000] 55:
[000] 56:
[000] 57:
[MEE] 58: function ZeigerWert(adr: FarAdr): ZeigerStr
;
[000] 59:
[VSA] 60: { Gibt Zeigerwert hexadezimal aus }
[000] 61:
[BH1] 62: begin
[PYC] 63: ZeigerWert := hex(seg(adr^)) + ':' + hex(
ofs(adr^));
[TU0] 64: end;
[000] 65:
[000] 66:
[G87] 67: begin { Hauptprogramm }
[2Y5] 68: writeln;writeln;
[1YG] 69: writeln('** Die PSP-Utility von DOS Inter
national...');
[922] 70: writeln;
[ZT2] 71: with PSP do
[BH1] 72: begin
[UVC] 73: writeln('Aufruf INT 20: ', hex(swap(i
nt20)));
[LAC] 74: writeln('Ende Speicher: ', hex(Segmen
t));
[6ND] 75: writeln('FarJump DOS: ', copy(hex(DO
S.Befehl),2,2),
[F77] 76: #32, ZeigerWert(DOS.Adr));
[HPD] 77: writeln('Ende-Adresse: ', ZeigerWe
rt(Ende));
[54F] 78: writeln('Abbruch-Adresse: ', ZeigerWe
rt(Abbr));
[TUF] 79: writeln('Fehler-Adresse: ', ZeigerWe
rt(Fehler));
[AMD] 80: writeln('Umgebungs-Segment: ', hex(Um
gebung));
[N93] 81: end; { with }
[5T0] 82: end.
```

Listing 3. Ein Programm zum Lesen des Program Segment Prefix in Turbo Pascal

```
[CU6] 10 REM EXE-Header-Utility
[624] 20 REM -----
[DT8] 30 REM (c) 1987 by DOS International
[3B1] 40 REM
[TG3] 50 PRINT:PRINT
```



```

[92G] 60 PRINT"*** Die EXE-Header-Utility von DOS Intern
ational..."
[9GA] 70 PRINT:INPUT"Name der EXE-Datei: ",DATEI$
[3MB] 80 IF DATEI$="" OR DATEI$="?" THEN FILES"*.exe":G
OTO 70
[BHB] 90 I=INSTR(DATEI$,"."):IF I THEN DATEI$=LEFT$(DAT
EI$,I-1)
[3V5] 100 DATEI$=DATEI$+".exe"
[LA4] 110 OPEN "r",#1,DATEI$
[H07] 120 FIELD #1, 128 AS FELD$:GET #1:Z=1
[5AA] 130 GOSUB 300:PRINT"Kennzeichen: ";MKI$(N)
[5AF] 140 GOSUB 300:PRINT"Größe der letzten Seite: ";N;"
Bytes"
[SFF] 150 GOSUB 300:PRINT"Dateigröße: ";N;" Blöcke zu 51
2 Bytes"
[N7E] 160 GOSUB 300:PRINT"Anzahl Relokationseinträge: ";
N
[9UD] 170 GOSUB 300:PRINT"Headergröße: ";N;" Paragraphs"
[CZC] 180 GOSUB 300:PRINT"Speicher min.: ";N;" Paragraph
s"
[PXC] 190 GOSUB 300:PRINT"Speicher max.: ";N;" Paragraph
s"
[XSA] 200 GOSUB 320:PRINT"Stapelsegment (SS): ";N$
[9HA] 210 GOSUB 320:PRINT"Stapelzeiger (SP): ";N$
[23B] 220 GOSUB 300:PRINT"negative Prüfsumme: ";N
[1NA] 230 GOSUB 320:PRINT"Befehlszeiger (IP): ";N$
[RCA] 240 GOSUB 320:PRINT"Codesegment (CS): ";N$
[C5E] 250 GOSUB 300:PRINT"Relokationstabelle: ";N;" (Off
set)"
[VCB] 260 GOSUB 300:PRINT"Nummer des Overlays: ";N
[252] 270 CLOSE #1
[VN1] 280 END
[7S1] 290 REM
[KBB] 300 N=ASC(MID$(FELD$,Z,1))+ASC(MID$(FELD$,Z+1,1))
*256:Z=Z+2
[R02] 310 RETURN
[L78] 320 N$=HEX$(CVI(MID$(FELD$,Z,2)))+N$:"H":Z=Z+2
[5Y6] 330 WHILE LEN(N$)<5:N$="0"+N$:WEND
[X02] 340 RETURN

```

Listing 4. Ein Basic-Programm zum Lesen des EXE-Headers

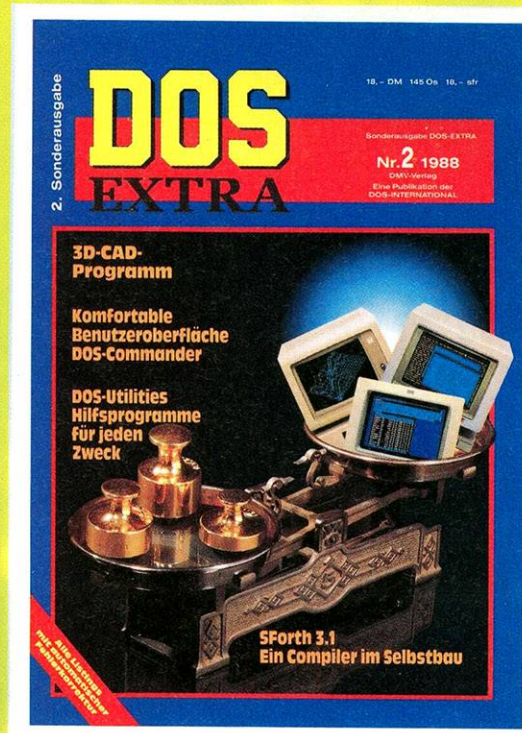
```

[BGR] 0: { Beispielprogramm für die Verwendung von "
Command.com" in Turbo Pascal }
[000] 1:
[9W4] 2: PROGRAM Command;
[000] 3:
[P23] 4: VAR ShellStr,
[TH6] 5: CurrComm : STRING[255];
[VXE] 6: Regs : RECORD ax,bx,cx,dx,bp,si
,di,ds,es,fi: INTEGER;
[TM0] 7: END;
[000] 8:
[000] 9:
[4U6] 10: { DOS-Befehl ausführen }
[RB5] 11: PROCEDURE DoCommand;
[BV7] 12: VAR Block: ARRAY[0..7] OF INTEGER;
[B21] 13: BEGIN
[000] 14:
[ZU8] 15: CurrComm := '/c '+CurrComm+RET;
[KTD] 16: CurrComm := Chr(Length(CurrComm)-1)+CurrC
omm;
[000] 17:
[Y19] 18: { Aufbau des Parameterblocks:
[VZG] 19: 1. Wort: 0 = Umgebung von diesem
Programm kopieren
[HK9] 20: 2. Wort: Offset des Kommandos
[4GA] 21: 3. Wort: Segment des Kommandos
[VDF] 22: 4. bis 7. Wort: für Dateien, hier nicht
gebraucht }
[TC3] 23: Block[0] := 0;
[FW7] 24: Block[1] := Of(CurrComm)+1;
[HF7] 25: Block[2] := Seg(CurrComm);
[000] 26:
[8FE] 27: Regs.ax := $4B00; { AX = Funkt
ionsnummer }
[MW7] 28: Regs.dx := Of(ShellStr)+1;
[BYH] 29: Regs.ds := Seg(ShellStr); { DS:DX = Ad
resse des Dateinamen }
[V37] 30: Regs.bx := Of(Block[0]);
[XWG] 31: Regs.es := Seg(Block); { ES:BX = Ad
resse des Parameters }
[MY2] 32: MsDos(Regs);
[TM0] 33: END;
[000] 34:
[000] 35:
[B21] 36: BEGIN
[4KA] 37: { Genauer Pfad für "command.com" }
[RV9] 38: ShellStr := 'c:\command.com'+Chr(0);
[000] 39:
[2W3] 40: WHILE TRUE DO BEGIN
[586] 41: Write ('DOS-Kommando: ');
[D45] 42: ReadLn (CurrComm);
[RY2] 43: DoCommand;
[TM0] 44: END;
[5L0] 45: END.

```

Listing 5. Beispielprogramm für die Verwendung von Comand.com

NICHT MIT GOLD AUFZUWIEGEN DOS EXTRA Nr. 2



18,- DM (Best.-Nr. 354)

EXTRA DOS-Commander: Fensterorien-
tierte Benutzeroberfläche für
MS-DOS

EXTRA Seitenweise Utilities, die den
Umgang mit dem PC erleichtern.

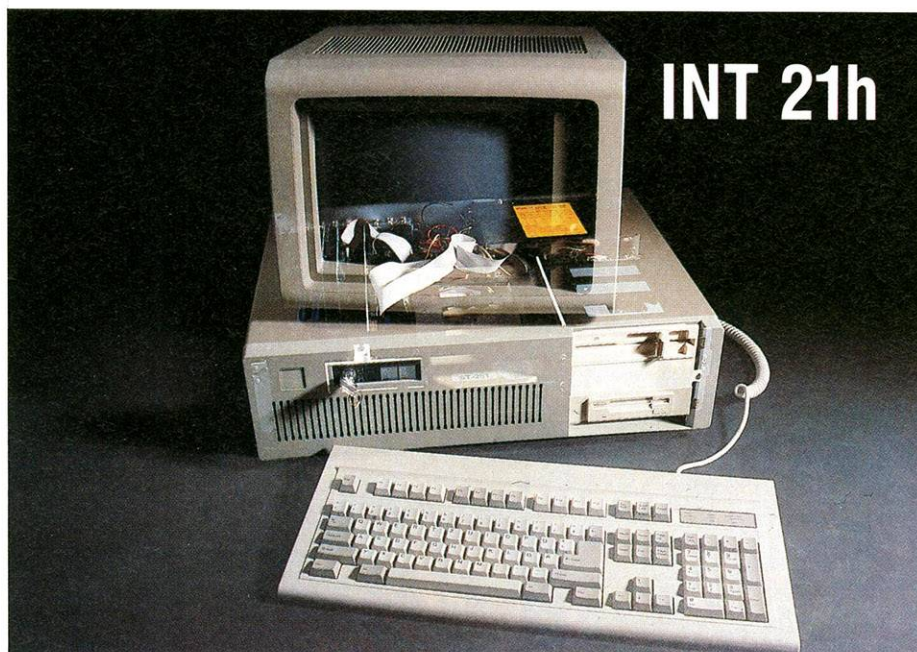
EXTRA Mit den SForth-Compiler 3.1 pro-
grammieren Sie in einer neuen
Dimension.

EXTRA Action, Spaß und Spannung:
Superspiele zum Abtippen

EXTRA 3D-CAD-Programm
der Superlative.

noch erhältlich bei

DMV · Postfach 250 · 3440 Eschwege
Bitte Bestellkarte benutzen



Der PC - glasklar (Teil 2)

Mit den Grundkenntnissen des ersten Teils können Sie nun etwas tiefer ins Innenleben des PC gehen. Um die Geschwindigkeit und Leistungsfähigkeit des Computers voll zu nutzen, lernen Sie die Grundfunktionen des BIOS kennen und sehen, wie sich diese Funktionen in Ihren eigenen Programmen verwenden lassen.

Alle Betriebssystem-Funktionen eines PC werden über die sogenannten Interrupts gesteuert. Dabei handelt es sich, wie es der Name bereits andeutet, um Unterbrechungen des laufenden Programms. Diese Unterbrechungen können entweder von der PC-Hardware oder von der Software, also vom laufenden Programm, ausgelöst werden. Im Prinzip verhält sich ein Interrupt ähnlich einem Unterprogrammaufruf: Der Prozessor unterbricht das laufende Programm, merkt sich die aktuelle Adresse des Programmzählers auf dem Stack und springt zur Interrupt-Routine. Nach Beendigung dieser Routine wird die Programmausführung an der gemerkten Adresse fortgesetzt.

Der 8088/8086-Prozessor stellt insgesamt 256 verschiedene Interrupts zur Verfügung. Jeder Interrupt wird durch seine Nummer, eine Zahl zwischen 0 und 255, identifiziert. Wie Sie bereits vom ersten Teil dieses Kurses wissen, bestehen Speicheradressen beim PC aus zwei einzelnen 16-Bit-Adressen, nämlich der Segment- und der Offset-Adresse. Während die Segment-Adresse ein 64-KByte-Segment innerhalb des 1-MByte-Adreßraumes bestimmt, ent-

hält der Offset-Teil die relative Adresse innerhalb dieses Segmentes. Eine komplette Adresse benötigt also vier Byte Speicherplatz, nämlich je zwei für Segment- und Offset-Teil. Die Interrupts stellen nun eine bequeme und kurze Methode dar, um beliebige Unterprogramme innerhalb des 1-MByte-Adreßbereichs aufzurufen.

Das funktioniert so: Zu jeder Interrupt-Nummer gibt es einen sogenannten Interrupt-Vektor. Ein solcher Interrupt-Vektor ist nichts anderes als eine Speicherstelle, die eine aus Segment- und Offset bestehende Adresse enthält. Die Offset-Adresse befindet sich in den ersten beiden Bytes des Vektors, die Segment-Adressen in den beiden folgenden Byte.

Für die Interrupt-Vektoren sind die ersten 1024 Byte des PC-Speichers reserviert. Dieser Bereich wird häufig die Interrupt-Vektortabelle genannt. Die Anzahl errechnet sich aus den insgesamt 256 verschiedenen Interruptnummern und den vier Byte, die jeder Vektor benötigt. Bild 1 verdeutlicht den Aufbau der Interrupt-Vektortabelle und ihre Lage im Hauptspeicher. Listing 1 zeigt ein kleines Basic-Programm zum Auslesen und Anzeigen aller Interrupt-Vektoren Ih-

res PC. Das Programm nimmt auch gleich eine grobe und einfache Klassifizierung der Interrupts vor. Diese Klassifizierung deutet an, von welcher Software der jeweilige Interrupt wahrscheinlich benutzt wird. Für diejenigen, die das Programm nicht eintippen wollen, zeigt Tabelle 1 alle beim PC fest verwendeten Interrupts. Die Abarbeitung eines Interrupts durch den Prozessor geschieht folgendermaßen: Als erstes wird der aktuelle Prozessorstatus gerettet, indem der Inhalt des Flag-Registers auf dem Stack abgelegt wird. Außerdem wird das aktuelle Codesegment (CS) und der aktuelle Programmzeiger (IP) auf den Stack gesichert. Diese Maßnahme dient dazu, die Programmausführung nach dem Ende der Interrupt-Routine wieder an der gleichen Stelle aufzunehmen, wo sie unterbrochen wurde. Bild 2 veranschaulicht die Verhältnisse auf dem Prozessorstack während des Aufrufs eines Interrupts. Das Interrupt-Flag wird nun gelöscht, um etwaige weitere Interrupts während der Bearbeitung des aktuellen Interrupts zu verhindern. Aus der Interrupt-Nummer errechnet die interne Prozessor-Logik die Adresse des zugehörigen Interrupt-Vektors. Aus diesem Vektor werden Register CS und IP neu geladen. CS:IP bilden zusammen den effektiven Programmzeiger. Das Laden dieses Register mit dem Inhalt des Interrupt-Vektors bedeutet also nichts anderes, als daß die Programmausführung mit der entsprechenden Interrupt-Routine fortgesetzt wird. Für die Beendigung solcher Interrupt-Routinen gibt es einen speziellen Assembler-Befehl, nämlich »iret«. Während der normale Befehl »ret« nur den Programmzeiger CS:IP vom Stack zurückholt und damit das Hauptprogramm fortsetzt, macht »iret« noch mehr: Hierdurch wird zusätzlich das Flag-Register wieder restauriert und das Interrupt-Flag gesetzt. Da alle Interrupt-Routinen mit diesem Befehl »iret« enden, ist es nicht möglich, beispielsweise mit dem Basic-Befehl »call« eine solche Interrupt-Routine aufzurufen. Der Call-Befehl produziert nämlich einen Aufruf eines einfachen Unterprogramms, rettet also das Flag-Register nicht auf den Stack. Der Befehl »iret« am Ende einer Interrupt-Routine kümmert sich darum aber nicht: Nachdem die Rückkehradresse in CS:IP geladen ist, holt der iret-Befehl das, was er für den Inhalt des Flag-Registers hält, vom Stack und führt anschließend den Rücksprung durch. Dadurch wird das Flag-Register mit einem zufälligen Wert geladen und gleichzeitig wird der Stack der aufrufenden Prozedur zerstört, da durch »iret« ein Wort zuviel vom Stack ab-

genommen wurde. Die Folge wird in den meisten Fällen ein völliger Absturz des PC sein.

Es gibt beim PC insgesamt sieben verschiedene Verwendungen der Interrupt-Vektoren:

- CPU-Interrupts: Diesen sind die ersten vier Interrupt-Vektoren zugeordnet. Sie werden bei bestimmten Betriebszuständen (Hardware-Fehler, Division durch Null und so weiter) des Prozessors aktiviert.
- Hardware-Interrupts: Diese Art von Interrupts wird durch die festverdrahtete PC-Hardware ausgelöst. Dazu zählen beispielsweise der Systemtakt-Interrupt, der Tastatur-Interrupt oder der Disketten-Controller-Interrupt.
- BIOS-Interrupts: Eine ganze Reihe von Interrupts ist dem ROM-residenten BIOS (also dem Betriebssystem-Kern) zugeordnet. Sowohl MS-DOS als auch Anwenderprogramme benutzen die BIOS-Interrupts, um »Low-Level« Systemdienste in Anspruch zu nehmen.

Adresse			Interrupt- Nummer
0:0	Offset ₀	Segment ₀	0
0:4	Offset ₁	Segment ₁	1
0:8	Offset ₂	Segment ₂	2
0:C	Offset ₃	Segment ₃	3
⋮	⋮	⋮	⋮
0:3F8	Offset _{FE}	Segment _{FE}	FE
0:3FC	Offset _{FF}	Segment _{FF}	FF

Adresse des Vektors = 0:(Interrupt-Nr. x 4)

Bild 1. Aufbau der Interrupt-Vektortabelle

Nummer dez	hex	Funktion
0	00	CPU-Fehler: Division durch Null
1	01	Single-Step-Interrupt (wird von »debug« verwendet)
2	02	Parity Error (NMI)
3	03	Break-Point-Interrupt (wird von »debug« verwendet)
4	04	CPU-Fehler: Arithmetischer Überlauf
5	05	Hardcopy: Wird bei Betätigen der PrtSc-Taste erzeugt
6	06	Schneider PC: Tastenabfrage Maus andere PCs: reserviert für Hardware
7	07	reserviert für Hardware
8	08	System-Uhr-Interrupt (Timer-Baustein)
9	09	Tastatur-Interrupt (wird bei Tastenanschlägen erzeugt)
10	0A	reserviert für Hardware (Echtzeituhr)
11	0B	reserviert für Hardware (2. serielle Schnittstelle)
12	0C	reserviert für Hardware (1. serielle Schnittstelle)
13	0D	Festplatten-Controller-Interrupt
14	0E	Disketten-Controller-Interrupt
15	0F	Drucker-Interrupt (parallele Schnittstelle)
16	10	BIOS-Bildschirmroutinen
17	11	BIOS-Ausstattungs-Routine
18	12	BIOS-Speicherbereichsabfrage
19	13	BIOS-Disketten- bzw. Festplattenroutinen
20	14	BIOS-RS232C-Routinen
21	15	IBM PC: BIOS-Kassettenrekorder-Routinen Schneider PC: Erweiterte BIOS-Funktionen andere PCs: nicht verwendet
22	16	BIOS-Tastaturroutinen
23	17	BIOS-Druckerroutinen
24	18	IBM PC: ROM-Basic aktivieren andere PCs: nicht verwendet
25	19	BIOS-System-Warmstart
26	1A	BIOS-Zeit- und Datums-Routinen
27	1B	Tastatur-Break-Interrupt (reserviert für Benutzer)
28	1C	Ticker-Interrupt (reserviert für Benutzer)
29	1D	Zeiger auf Tabelle mit Bildschirmparametern
30	1E	Zeiger auf Tabelle mit Laufwerksparametern
31	1F	Zeiger auf Tabelle mit Bildschirm-Grafikzeichen
32	20	MS-DOS-Programmende-Routine
33	21	MS-DOS-Funktionsinterrupt
34	22	MS-DOS-Programmende-Folgeroutine
35	23	MS-DOS-Ereignisbehandlung Ctrl-C
36	24	MS-DOS-Ereignisbehandlung kritischer Fehler
37	25	MS-DOS-Routine Diskette absolut lesen
38	26	MS-DOS-Routine Diskette absolut schreiben
39	27	MS-DOS-Programm beenden, aber im Speicher verbleiben
40	28	»MS-DOS-Frei« Interrupt
51	33	Maus-Routinen
73	49	Zeiger auf Umkodierungstabelle für mit der Tastatur verbundene Geräte
103	67	Funktionen für erweiterten Speicher

Tabelle 1. Die wichtigsten Interrupts beim PC

- MS-DOS-Interrupts: Hierbei handelt es sich um Interrupts, die vom Betriebssystem (MS-DOS oder PC-DOS) verwendet werden. Am wichtigsten ist der Interrupt 21hex, durch den Anwenderprogramme die MS-DOS-Funktionen aufrufen können.
- Basic-Interrupts: Da beim IBM-PC das Basic zumindest teilweise im ROM abgelegt ist, werden ähnlich wie für das BIOS einige Interrupts von Basic verwendet. Beim Verlassen von Basic werden diese Interrupts wieder freigegeben.
- Adressen von Tabellen: Hierbei handelt es sich um gar keine Interrupts im eigentlichen Sinne. Die

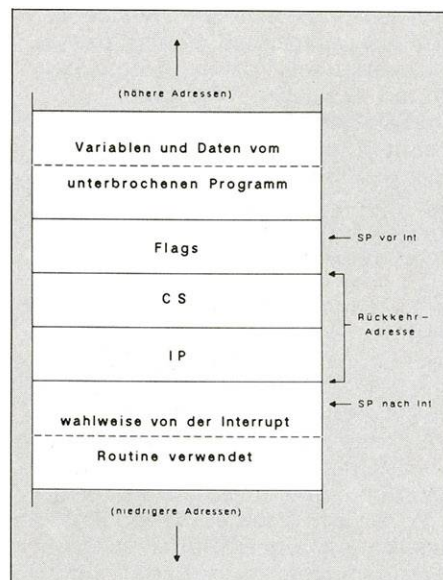


Bild 2. Der Prozessorstack beim Interrupt-Aufruf

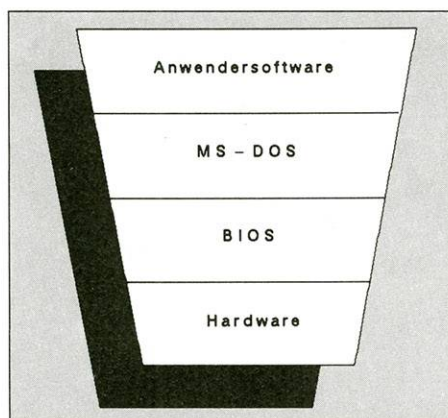


Bild 3. Die Software-Hierarchie beim PC

sogenannten Adreß-Interrupts enthalten keinen Zeiger auf eine Interrupt-Routine, sondern zeigen auf spezielle Tabellenbereiche mit globaler Bedeutung, beispielsweise auf Tabellen mit Bildschirm- oder Laufwerkparametern.

- Anwender-Interrupts: Die Interrupts 96 bis 103 sind für temporäre Nutzung durch Anwenderprogramme reserviert. In der Regel benutzen Programme aber auch den Bereich von 128 bis 255, der eigentlich für die Verwendung durch MS-DOS und Basic reserviert ist. Dieses Verfahren ist problemlos, solange das Programm vor der Belegung eines Interrupt-Vektors prüft, ob er noch frei ist (Inhalt 0000:0000).

Das BIOS ist der harte Kern des PC-Betriebssystems und sorgt für die Kompatibilität

PC-Software läuft in praktisch unveränderter Form auf IBM-kompatiblen PCs ganz verschiedener Hersteller und mit unterschiedlichen Hardware-Ressourcen. Vom ganz einfachen PC mit 128 KByte RAM und 8088-Prozessor und 4,77 MHz Taktfrequenz geht die Spannbreite bis zum 80386-AT mit 8 MByte Speicher, einer 32-Bit-CPU und 20 MHz Taktfrequenz. Daneben gibt es verschiedene Grafikkarten wie CGA, EGA, VGA oder Hercules, Laufwerke mit 360 KByte und solche mit 1,2 MByte und Festplatten ganz unterschiedlicher Größenordnungen. Eigentlich grenzt es an ein Wunder, wenn trotz dieser großen Unterschiede in der Hardware dennoch die überwiegende Mehrzahl aller Anwender-Programme problemlos auf jedem x-beliebigen PC läuft. Doch in diesem Fall hat das Wunder einen Namen, oder besser gesagt gleich drei: Das gesamte Betriebssystem eines PC besteht nämlich aus drei grundlegenden Kompo-

nenten: Dem BIOS, dem MS-/PC-DOS und dem Kommandoprozessor »command.com«, der bereits im ersten Teil dieses Kurses ausführlich behandelt wurde. Diese drei Komponenten bilden praktisch eine Hierarchie, welche die Anwenderprogramme vor der Hardware abschirmt (Bild 3) und eine gleichartige Benutzeroberfläche für eigentlich recht verschiedene Computersysteme zur Verfügung stellt.

Die unterste Ebene dieser Hierarchie bildet das BIOS. Es ist auch schon der einzige hardware-abhängige Teil des gesamten Betriebssystems. Für diejenigen, die es noch nicht wissen, die vier Buchstaben BIOS sind die Abkürzung für »Basic Input-Output System«, deutsch etwa »Basissystem für Ein- und Ausgabe«. Das BIOS ist in der Hardware implementiert, und steht deshalb immer zur Verfügung, ohne daß es geladen werden muß. Es besteht aus einer Menge Routinen, die alle notwendigen Hardware-Funktionen anbieten. Alle anderen Komponenten sind hardware-unabhängig, wenn man einmal davon absieht, daß der Prozessor selbst kompatibel zum Intel 8086/8088-Prozessor sein muß, damit die Software überhaupt laufen kann. Hardware-unabhängig bedeutet, daß die Programme nicht direkt mit der Hardware umgehen müssen, um (zum Beispiel) ein Zeichen auf den Bildschirm zu bringen. Stattdessen wird die entsprechende BIOS-Routine aufgerufen.

Das BIOS wird in der Regel vom Hersteller des Computers entwickelt oder eingekauft und angepaßt. Es befindet

Wert Bedeutung

0	kein Fehler
1	ungültiger Befehl
2	Adreßmarke nicht gefunden
3	Diskette ist schreibgeschützt
4	Sektor nicht gefunden
8	DMA-Fehler
16	Prüfsummenfehler
32	Fehler im Disketten-Controller
64	Spur nicht gefunden
128	Laufwerk nicht bereit

Tabelle 3. Fehlerstatus der Disketten-Routinen

sich fest installiert in einem speziellen Speicherbereich (»Read Only Memory« ROM) und wird beim Einschalten des Computers aktiviert. Je nach vorhandener Hardware und Computertyp (PC, XT, AT, 80386er etc.) fällt das BIOS immer etwas anders aus. Vorbild für alle BIOS-Versionen ist dabei das IBM-BIOS, allerdings wacht IBM eifersüchtig darüber, daß kein Hersteller etwa das IBM-BIOS einfach »abkupfert«.

Sinn und Zweck des BIOS ist es, die unmittelbare Kommunikation mit der Hardware zu übernehmen. Die Philosophie dahinter ist die, daß Anwendersoftware grundsätzlich nicht direkt die Hardware ansprechen sollten. Daher enthält das BIOS bereits ausführliche und komfortable zu benutzende Routinen, um alle grundlegenden Computerfunktionen anzusprechen, zum Beispiel das Lesen und Schreiben von Disketten-Sektoren, die Ein- und Ausgabe von Zeichen oder Datums- und Zeitfunktionen. Die einzelnen Routinen werden über die bereits angesprochenen

Nr. (AH)	Funktion	Register	Eingabe	Ausgabe
0	Diskettensystem zurücksetzen		AH: Status	
1	Fehlerstatus holen		AH: Status	
2	Sektor lesen		DH: Seite (0..1) DL: Laufwerk (0..1) CH: Spur (0..39) CL: Sektor (1..9) ES:BX: Datenpuffer AL: Anzahl Sektoren (1..9)	AH: Status AL: Anzahl Sektoren
3	Sektor schreiben		DH: Seite (0..1) DL: Laufwerk (0..1) CH: Spur (0..39) CL: Sektor (1..9) ES:BX: Datenpuffer AL: Anzahl Sektoren (1..9)	AH: Status AL: Anzahl Sektoren CL: Sektor (1..9)
4	Sektor verifizieren		DH: Seite (0..1) DL: Laufwerk (0..1) CH: Spur (0..39) CL: Sektor (1..9) AL: Anzahl Sektoren (1..9)	AH: Status AL: Anzahl Sektoren
5	Spur formatieren		DH: Seite (0..1) DL: Laufwerk (0..1) CH: Spur (0..39) ES:BX: Formatpuffer	AH: Status

Der Formatpuffer muß für jeden Sektor vier Bytes an Informationen enthalten: (1) Spurnummer, (2) Diskettenseite, (3) Sektornummer, (4) Sektorgroße codiert als: 0 für 128 Byte, 1 für 256 Byte, 2 für 512 Byte, 3 für 1024 Byte.

Tabelle 2. Unterfunktionen des BIOS-Interrupts 19

Funktion	Nummer	Eingabe	Ausgabe
Ein Zeichen drucken	0	AL: Zeichen	AH: Status
Drucker initialisieren	1	—	AH: Status
Druckerstatus abfragen	2	—	AH: Status

Tabelle 4. Unterrouinen des BIOS-Interrupts 23

BIOS-Interrupts aktiviert. Wie die jeweiligen BIOS-Routinen im einzelnen ihre Aufgaben erledigen, interessiert dabei überhaupt nicht mehr.

Eine BIOS-Routine wird wie eine Black Box behandelt: Man kennt den Eingang und weiß, was der Aufruf als Ergebnis liefert; alles andere ist uninteressant. Dadurch wird erreicht, daß Änderungen der Hardware völlig problemlos und vor allem ohne neue Anpassung der Anwender-Software möglich sind. Es werden vom Hersteller einfach die entsprechenden BIOS-Routinen neu geschrieben, die die geänderte Hardware ansprechen. Da aber die geänderte Routine nach wie vor über den gleichen Interrupt und mit gleicher Register-Belegung aufgerufen wird, gibt es aus der Sicht des Anwenderprogramms gar keinen Unterschied.

Auch MS-DOS bedient sich grundsätzlich des BIOS, um seine Aufgaben zu erledigen. MS-DOS ist das eigentliche Betriebssystem, das höherwertige Funktionen für Anwenderprogramme zur Verfügung stellt. Da auch MS-DOS die Hardware über das BIOS anspricht, läuft das Betriebssystem im Idealfall unverändert auf vielen verschiedenen Computern.

Anwenderprogramme sollten sich daher der MS-DOS-Funktionen bedienen, um ihre Aufgaben zu erledigen. Viele Anwenderprogramme rufen BIOS-Routinen auch direkt auf und umgehen damit das MS-DOS. Damit wird zwar immer noch eine weitgehende Hardware-Unabhängigkeit erreicht, die Kommunikation mit dem BIOS spielt sich aber schon auf einer deutlich niedrigeren Ebene ab. Beispielsweise »weiß« das BIOS rein gar nichts von Dateien. Es kennt nur Disketten-Sektoren, die gelesen, geschrieben, formatiert oder verifiziert werden können. Das ganze Dateisystem ist eine Leistung des MS-DOS. Nun gibt es Anwendungen, wo gerade dieser »Low-Level« Aspekt des BIOS im Vordergrund steht. Man denke da nur an Programme wie ei-

Adresse (Hex)	Gerät
0040:0008	LPT1
0040:000A	LPT2
0040:000C	LPT3
0040:000E	LPT4

Tabelle 5. Speicherzellen für die E/A-Adressen der vier parallelen Schnittstellen

nen Disketten-Monitor oder ähnliche Utilities. Vielfach werden BIOS-Routinen auch ganz einfach aus Geschwindigkeitsgründen verwendet. Beispielsweise ist die Bildschirmausgabe über das BIOS deutlich schneller als über MS-DOS, vor allem, wenn noch ein zusätzlicher Bildschirmtreiber wie »ansi.sys« aktiviert ist.

Nach soviel Theorie nun wieder etwas Praxis: Natürlich können Sie die BIOS-Routinen auch für eigene Programme ausnutzen. Besonders einfach gestaltet sich das in Turbo Pas-

cal 3.0. Hier gibt es den Befehl »intr«, der den Aufruf des als Parameter angegebenen Interrupts ermöglicht. Der Aufruf erfolgt in der Form

intr(nr, reg)

wobei »nr« die Interrupt-Nummer ist und »reg« eine Datenstruktur, die die Prozessorregister darstellt. Vor dem Aufruf der Interrupt-Prozedur kann man durch Zuweisungen an »reg« praktisch die Prozessorregister auf die Werte einstellen, die an die BIOS-Routine übermittelt werden sollen. Nach dem Aufruf kann man umgekehrt eventuelle Ergebnisse auslesen. Listing 2 zeigt ein einfaches Beispielprogramm zur Nutzung des »intr«-Befehls.

Bei der Benutzung von BIOS-Routinen ist äußerste Vorsicht geboten. Jeder Programmierer, der die »intr«-Prozeduren in eigenen Programmen nutzen möchte, sollte sich unbedingt

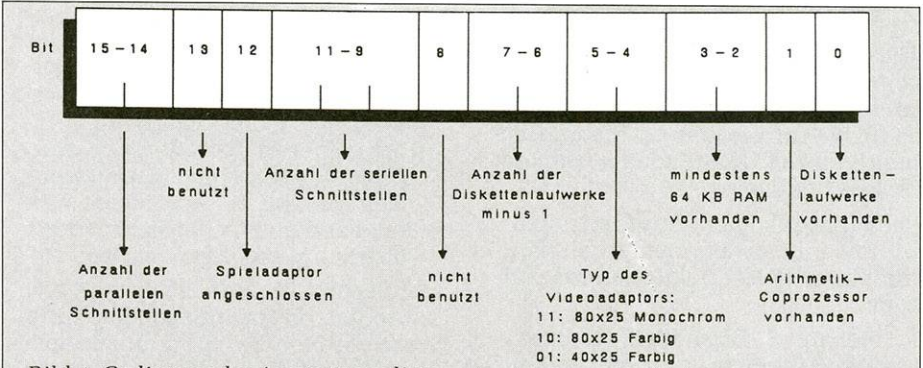


Bild 4. Codierung der Ausstattungsliste

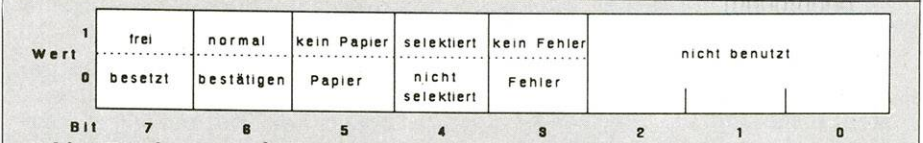
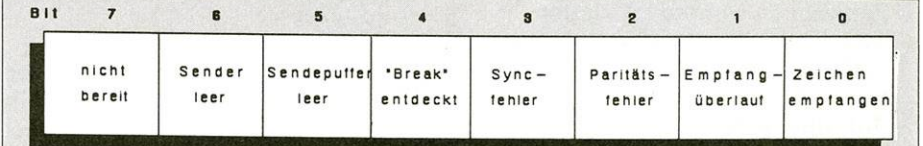


Bild 5. Druckerstatus des Interrupts 23



Jedes Bit hat die Bedeutung im positiven Sinn (wie oben beschriftet) wenn das Bit »1« ist.

Bild 6. Leitungs- und Modemstatus der RS232C-Routinen

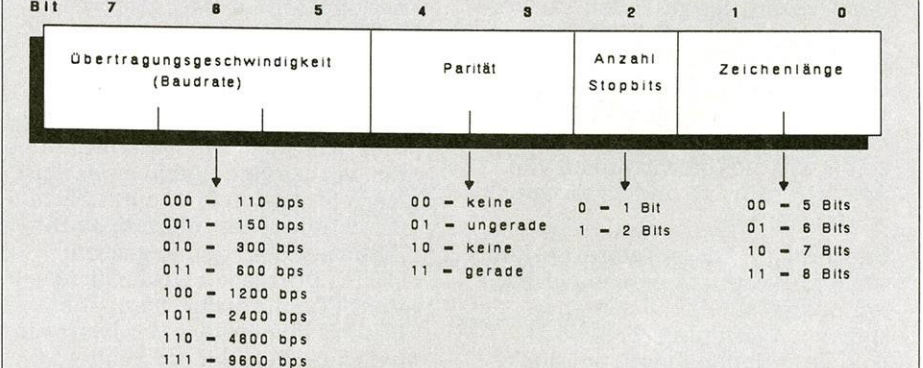


Bild 7. Das Parameterbyte für die RS232C-Parameter

Nr. (AH)	Funktion	Register Eingabe	Ausgabe
0	RS232C-Parameter setzen	DX: Kanalnummer (0..1) AL: Parameterbyte (Bild 7)	AX: Status
1	Zeichen senden	DX: Kanalnummer (0..1) AL: Zeichen	AX: Status
2	Zeichen empfangen	DX: Kanalnummer (0..1)	AH: Status AL: Zeichen
3	Kanalstatus holen	DX: Kanalnummer (0..1)	AX: Status

Tabelle 6. Unterfunktionen des BIOS-Interrupts 20

zumindest mit den Grundlagen der Assembler-Programmierung und der PC-Hardware beschäftigen. Im übrigen gelten die gleichen Grundregeln wie für jede Form der maschinennahen Programmierung: Starten Sie ein solches selbstgeschriebenes oder abgetipptes Programm niemals, ohne es zuvor auf Diskette gesichert zu haben. Bei fehlerhafter Programmierung besteht sonst die Gefahr, daß Ihr PC unrettbar »abstürzt«. Arbeiten Sie grundsätzlich nur mit den Sicherheitskopien Ihrer Programm- und Datendisketten, denn durch fehlgehende BIOS-Aufrufe können Sie unbeabsichtigt Ihre Diskettendaten irreparabel beschädigen.

Nach dieser langen Vorrede wollen wir uns nun etwas genauer mit den einzelnen BIOS-Funktionen beschäftigen.

- Interrupt 5: Diese BIOS-Routine veranlaßt einen Ausdruck des aktuellen Bildschirminhalts auf dem angeschlossenen Drucker. Eine Betätigung der Taste [PrtSc] löst den Interrupt 5 aus.
- Interrupt 16: Dieser BIOS-Interrupt stellt insgesamt 16 Unterfunktionen rund um die Bildschirmansteuerung zur Verfügung. Diese Funktionen werden im vierten Teil dieser Reihe ausführlich behandelt.
- Interrupt 17: Diese BIOS-Routine stellt die Ausstattung des PC fest und meldet diese in Form einer bit-codierten Tabelle im Register AX zurück (Bild 4).
- Interrupt 18: Diese Routine ergibt den Umfang des gesamten zur Verfügung stehenden Schreib-Lese-speichers in KByte im AX-Register. Bei vollem Speicherausbau wird der Wert 640 geliefert. Das Programm »Konfig.bas« in Listing 3 zeigt, wie die Informationen von den Interrupts 17 und 18 ausgewertet werden.
- Interrupt 19: Dieser Interrupt stellt sechs Unterfunktionen zum direkten Zugriff auf das Diskettensystem zur Verfügung (Tabelle 2). Alle Funktionen liefern bei Rückkehr ein Statusbyte im AH-Register. Tabelle 3 zeigt die Bedeutung der einzelnen Werte.
- Interrupt 20: Hier sind vier Unterfunktionen zum Ansprechen der RS232C-Schnittstelle vorhanden. Diese Funktionen werden im nächsten Abschnitt dieses Artikels beschrieben.
- Interrupt 21: Dieser Interrupt wird bei den PCs nicht einheitlich verwendet. Der IBM-PC enthält an dieser Stelle Routinen zur Steuerung eines externen Kassettenrecorders (!) für Zwecke der Programmaufzeichnung. Beim Schneider-PC befinden sich an dieser Stelle die sogenannten erweiterten BIOS-Funktionen, während andere PCs diesen Interrupt überhaupt nicht verwenden.
- Interrupt 22: Durch Interrupt 22 werden die grundlegenden Tastaturroutinen des BIOS aufgerufen. Insgesamt drei Unterfunktionen sind vorhanden, die im nächsten Kursteil ausführlich behandelt werden.
- Interrupt 23: Durch ebenfalls drei Unterfunktionen wird hier der Drucker am Centronics-Port angesprochen. Diese Funktionen werden später in diesem Artikel beschrieben.
- Interrupt 24: Dieser Interrupt aktiviert beim IBM-PC das ROM-Basic, bei kompatiblen PCs wird in der Regel ein System-Neustart ausgelöst.
- Interrupt 25: Der Aufruf des Interrupts 25 hat die gleiche Wirkung wie das Drücken der Tastenkombination [Ctrl-Alt-Del]: Das System wird neu gebootet.
- Interrupt 26: Hier befinden sich die Datums- und Uhrzeitroutinen des BIOS. Bei einfachen PCs sind hier lediglich zwei Unterfunktionen vorhanden, nämlich das Auslesen (Funktion 0) und das Setzen (Funktion 1) des internen 32-Bit-Zählers, der in den Registern CX:DX übergeben wird. Die meisten ATs und einige kompatible PCs mit eingebauter Hardware-Uhr (wie zum Beispiel der Schneider-PC) besitzen zusätzliche Routinen

Adresse (Hex)	Gerät
0040:0000	COM1
0040:0002	COM2
0040:0004	COM3
0040:0006	COM4

Tabelle 7. Speicherzellen für die E/A-Adressen der vier seriellen Schnittstellen

zum Lesen und Setzen von Uhrzeit, Datum und Alarmzeit. Soweit unser Schnellüberblick über die BIOS-Routinen des PC. Alle wichtigen BIOS-Funktionen stehen Ihnen durch die BIOS-Bibliothek im Listing 4 unter Turbo Pascal zur Verfügung – Experimentieren ist ausdrücklich erlaubt! Nun, und in weiteren Teilen dieses Kurses, erfahren Sie mehr über die spezifischen Gruppen der BIOS-Routinen und die dazugehörige Hardware.

Die parallele Schnittstelle nach dem Centronics-Standard sorgt beim PC für die problemlose Drucker-Verbindung

Die BIOS-Funktionen der Druckersteuerung sind in Tabelle 4 aufgelistet. Die Funktionen 0 und 1 sind einfach und erklären sich selbst; die Funktion 2 gibt ein Statusbyte zurück, dessen Aufbau in Bild 5 grafisch dargestellt wird. Als Beispiel für die Verwendung des Statusbytes mag das kleine Programm in Listing 5 dienen. Das Programm verwendet die BIOS-Drucker-Funktion 2 um abzufragen, ob der Drucker betriebsbereit ist. Wenn dies der Fall ist, gibt das Programm einen Returncode gleich 0 zurück. Falls der Drucker nicht betriebsbereit ist, gibt das Programm eine »1« zurück. Mit diesem Programm und der Stapeldatei »druck.bat« in Listing 6, können Sie den ärgerlichen »Schreibfehler beim Gerät PRN« ein für alle Mal überwinden.

Eine Erweiterung zum BIOS sollte auch erwähnt werden. Der PC ist grundsätzlich in der Lage, bis zu vier parallele Schnittstellen zu betreiben. Die Schnittstellen unterscheiden sich durch verschiedene Ein- und Ausgabeadressen, sind aber sonst vom MS-DOS oder vom BIOS gesehen identisch und werden sogar durch die gleichen Routinen gesteuert. Ein Problem taucht jedoch auf, wenn man zwei Drucker angeschlossen hat. Es ist in den meisten Fällen so gut wie unmöglich, einem Programm mitzuteilen, daß es jetzt bitte einen ande-

ren Drucker verwenden soll. Die meisten Programme sind nämlich fest auf »lpt1« eingestellt oder fragen nur während der Installation, welcher Drucker verwendet werden soll. Der Mode-Befehl des MS-DOS ist auch nicht in der Lage, einen Drucker auf einen anderen umzuleiten, wenn beide Drucker an parallelen Schnittstellen angeschlossen sind. Was kann man da nur machen?

Hier hilft indirekt wieder das BIOS. Es holt sich nämlich die Adresse für jede parallele Schnittstelle aus einer Tabelle, die es beim Einschalten des PC im Speicher aufbaut. Diese Tabelle dürfen Sie ändern mit dem erfreulichen Ergebnis, daß das BIOS einfach die geänderte Adresse liest und verwendet. Damit ist die Ausgabe in der Praxis auf einen anderen Drucker umgeleitet. In Tabelle 5 sind die Tabellenadressen für alle vier Drucker zu finden. Um zum Beispiel die Ausgabe von »lpt1« auf »lpt2« umzuleiten, macht man in GW-Basic einfach folgendes:

```
def seg = &H40
AltLpt1 = peek(&H8) + 256 * peek(&H9)
poke &H8, peek(&H0A)
poke &H9, peek(&H0B)
def seg
```

In Turbo Pascal sieht das so aus:

```
AltLpt1 := MemW[$40:$8];
MemW[$40:$8] := MemW[$40:$A];
```

In beiden Beispielen wurde die Lpt1-Adresse in der Variablen »AltLpt1« gespeichert. Andere Umleitungen, beziehungsweise die Aufhebung der Umleitung, werden analog realisiert.

Die RS232-Schnittstelle – das Universalgenie für die serielle Kommunikation

In Tabelle 6 sind die BIOS-Routinen zur Steuerung der seriellen Schnittstelle aufgeführt. Diese Routinen melden im Register AX einen erweiterten Fehlercode zurück. Das Teilregister AH enthält dabei den Leitungstatus, das Teilregister AL den Modemstatus (Bild 6). Die Initialisierung der Schnittstelle erfolgt durch Unterfunktion 0, wobei im Register AL ein Parameterbyte übergeben wird. Bild 7 zeigt den Aufbau dieses Parameterbytes.

Was oben über die parallelen Schnittstellen gesagt wurde, gilt auch für die seriellen Schnittstellen. Auch hier

führt das BIOS eine Adressen-Tabelle, die zwecks Umleitung manipuliert werden kann. Tabelle 7 führt die notwendigen Auskünfte auf. Die Änderung der BIOS-Adressen-Tabelle erfolgt wie im Drucker-Beispiel oben.

Eine Einschränkung dieser Methode muß beachtet werden. Einfach durch Änderung einer Adresse können Sie nicht eine serielle auf eine parallele Schnittstelle (oder umgekehrt) umleiten. Die Logik der beiden Schnittstellen ist nämlich sehr unterschiedlich. Wie schon erwähnt wurde, ändern Sie mit der vorgestellten Methode nur die Adresse einer Schnittstelle, aber nicht die Zugriffslogik. Für die Umleitung von parallel auf seriell gibt es den Mode-Befehl des MS-DOS, der leider meistens nur zwei Schnittstellen jeder Art kennt.

Damit sind Sie am Ende des zweiten Teils unseres PC-Kurses angelangt. Auf den folgenden Seiten werden in weiteren Folgen noch mehr Geheimnisse der Hardware und des Betriebssystems offenbart – und je mehr Sie wissen, desto effektiver können Sie Ihren PC einsetzen!

(Michael Seyler/Thomas Little/ev)

```
[ABA] 10 REM Interrupt-Vektoren lesen
[1B1] 20 REM
[Al2] 30 DEF SEG=0
[3N6] 40 DEF FND(X)=PEEK(X)+256*PEEK(X+1)
[9N7] 50 DEF FNH$(X,N)=RIGHT$("0000"+HEX$(X),N)
[G43] 60 FOR I=0 TO 255
[HK9] 70 PRINT "INT ";PRINT USING"###";I;PRINT" ";
[BP4] 80 SEG$=FNH$(FND(I*4+2),4)
[2N4] 90 OFS$=FNH$(FND(I*4),4)
[C25] 100 PRINT SEG$;" ";OFS$;
[479] 110 IF SEG$>="FC00" THEN PRINT "BIOS":GOTO 170
[TCB] 120 IF SEG$>="A000" THEN PRINT "Zusatz-ROM":GOTO 170
[170]
[H7E] 130 IF SEG$>="2000" THEN PRINT "Anwender-Programm":GOTO 170
[589] 140 IF SEG$>="0200" THEN PRINT"MS-DOS":GOTO 170
[LL9] 150 IF SEG$>="0000" THEN PRINT"System":GOTO 170
[626] 160 PRINT"nicht belegt"
[VT1] 170 NEXT
```

Listing 1. Ein Basic-Programm zur Anzeige aller Interrupt-Vektoren

```
[WP5] 0: program HardCopy;
[000] 1:
[TR7] 2: { Bildschirm-Hardcopy }
[000] 3:
[A21] 4: type
[0XL] 5: register = record { varianter Record für
Prozessor-Register }
[W05] 6: case boolean of
[TRE] 7: true: (ax,bx,cx,dx,bp,si,d
i,ds,es,flags: integer);
[22B] 8: false: (al,ah,bl,bh,cl,ch,
dl,dh: byte);
[TU0] 9: end;
[000] 10:
[000] 11:
[TS5] 12: var reg: register;
```

```
[000] 13:
[000] 14:
[BH1] 15: begin
[8PA] 16: intr(5, reg); { Aufruf BIOS-Routine }
[TU0] 17: end;
[000] 18:
```

Listing 2. »Hardcopy« als Beispiel eines BIOS-Aufrufs mit Turbo Pascal

```
[G9F] 0: ' KONFIG.BAS - Konfigurationswort vom BIOS
auswerten
[3R4] 1: ' (Turbo Basic)
[000] 2:
[102] 3: DEFLNG a-z
[000] 4:
[1KK] 5: ' Symbolische Namen für die Register machen
alles einfacher
[EUI] 6: %FLAGS = 0
[H71] 7: %AX = 1
[N71] 8: %BX = 2
[T71] 9: %CX = 3
[X71] 10: %DX = 4
[G71] 11: %SI = 5
[M61] 12: %DI = 6
[771] 13: %BP = 7
[P71] 14: %DS = 8
[U71] 15: %ES = 9
[000] 16:
[89G] 17: ' Textbildschirm selektieren und Titel ausg
eben
[FV3] 18: SCREEN 0: WIDTH 80
[KN1] 19: COLOR 10
[349] 20: LOCATE 1,25: PRINT "PC Konfiguration"
[000] 21:
[7KL] 22: ' BIOS Interrupt 12H liefert Speichergröße
in KByte im AX-Register
[602] 23: LOCATE 5,15
[C14] 24: CALL INTERRUPT &H12
```



```

[3ME] 25: PRINT REG(%ax); "KByte Hauptspeicher installiert"
[000] 26:
[SYN] 27: ' BIOS Interrupt 11H liefert die Hardware-Konfiguration im AX-Register
[602] 28: LOCATE 7,15
[914] 29: CALL INTERRUPT &H11
[GY4] 30: Hardware = REG(%ax)
[000] 31:
[Z2C] 32: ' Anzahl Disketten-Laufwerke ausgeben
[BZ5] 33: IF (Hardware AND 1) <> 0 THEN
[G1A] 34: Disketten = (Hardware AND &H00C0) \ &H0040 + 1
[AYB] 35: PRINT Disketten; "Disketten Laufwerk(e)"
[Y91] 36: END IF
[000] 37:
[4PM] 38: ' Von der Konfiguration werden die restlichen Zusätze ausgegeben
[602] 39: LOCATE 9,15
[6BK] 40: PRINT ABS((Hardware AND &HC000) \ &H4000); "parallele(n) Schnittstelle(n)"
[862] 41: LOCATE 10,15
[HHM] 42: PRINT (Hardware AND &H0600) \ &H0200; "serielle(n) Schnittstelle(n)"
[000] 43:
[A62] 44: LOCATE 12,15
[WY5] 45: IF (Hardware AND 2) = 0 THEN
[D33] 46: PRINT "Kein";
[Y91] 47: END IF
[D0D] 48: PRINT "Arithmetik Coprozessor installiert"
[000] 49:
[C62] 50: LOCATE 14,15
[9F7] 51: IF (Hardware AND &H1000) = 0 THEN
[D33] 52: PRINT "Kein";
[Y91] 53: END IF
[6AA] 54: PRINT "Spieladapter angeschlossen"
[DD0] 55: END

```

Listing 3. »Konfig«: ein Turbo-Basic-Programm interpretiert die Hardware-Ausstattungs-Bytes des BIOS

```

[C3D] 0: { BIOS.INC -- Include-Datei für BIOS-Routinen }
[000] 1:
[7X7] 2: { (c) 1987 DOS International }
[000] 3:
[000] 4:
[HAF] 5: { Diese Datei enthält alle wichtigen und nicht }
[SRG] 6: { bereits durch Standard-Prozeduren abgedeckt }
[KPD] 7: { BIOS-Funktionen in Form von einfach zu verwenden }
[7DD] 8: { wendenden Prozeduren und Funktionen. }
[000] 9:
[7AC] 10: { Hinweise zur Anwendung im DOS Extra 3 }
[YE9] 11: { Kurs: "Der PC - glasklar (2)" }
[000] 12:
[000] 13:
[000] 14:
[FS7] 15: { ***** Vereinbarungen ***** }
[000] 16:
[000] 17:
[A21] 18: type
[9K5] 19: register = record
[W05] 20: case boolean of
[TRE] 21: true: (ax,bx,cx,dx,bp,si,di,ds,es,flags: integer);
[22B] 22: false: (al,ah,bl,bh,cl,ch,dh: byte);
[TU0] 23: end;
[000] 24:
[000] 25:
[EM0] 26: var
[3A4] 27: reg: register;
[000] 28:
[000] 29:
[000] 30:
[GN9] 31: { ***** Allgemeine Funktionen ***** }
[000] 32:
[000] 33:
[02B] 34: function Bit(b, n: integer): boolean;
[000] 35:
[CUE] 36: { ergibt Zustand von Bit b der Zahl n; dient zur }

```

```

[C9F] 37: { Abfrage Bit-codierter Status-Informationen }
[000] 38:
[BH1] 39: begin
[C49] 40: b := b and 15; { Bits 0..15 erlaubt }
[SA6] 41: Bit := ((n shr b) and 1) = 1;
[TU0] 42: end;
[000] 43:
[000] 44:
[000] 45:
[K1D] 46: function SystemConfiguration: integer;
[000] 47:
[HLD] 48: { ergibt Bit-codierte Ausstattungsliste }
[000] 49:
[Z26] 50: { Bit Bedeutung }
[Y75] 51: { ----- }
[G79] 52: { 0 Disklaufwerk vorhanden }
[5M6] 53: { 1 8087 installiert }
[ZW7] 54: { 2..3 (immer gesetzt) }
[7S7] 55: { 4..5 Bildschirm-Modus }
[ZU8] 56: { 6..7 Disklaufwerke über 1 }
[VV6] 57: { 8 kein DMA-Chip }
[WC7] 58: { 9..11 Anzahl RS232-Ports }
[RU7] 59: { 12 Spieleadapter }
[LG6] 60: { 13 (unbenutzt) }
[AV7] 61: { 14..15 Anzahl Drucker }
[000] 62:
[BH1] 63: begin
[6W8] 64: intr(17, reg); { Ausführung }
[LZ8] 65: SystemConfiguration := reg.ax;
[TU0] 66: end;
[000] 67:
[000] 68:
[000] 69:
[9M9] 70: function MemorySize: integer;
[000] 71:
[6PG] 72: { ergibt die verfügbare Speicherkapazität in n KByte }
[000] 73:
[BH1] 74: begin
[DW8] 75: intr(18, reg); { Ausführung }
[RZ5] 76: MemorySize := reg.ax;
[TU0] 77: end;
[000] 78:
[000] 79:
[000] 80:
[TB8] 81: function ShiftStatus: byte;
[000] 82:
[4CD] 83: { ergibt Statusbyte für Sondertasten }
[000] 84:
[6H6] 85: { Bit Wert Bedeutung }
[K04] 86: { ----- }
[6K5] 87: { 0 1 Right Shift }
[KN5] 88: { 1 2 Left Shift }
[FY3] 89: { 2 4 Ctrl }
[BU3] 90: { 3 8 Alt }
[ZE5] 91: { 4 16 Scroll Lock }
[FX4] 92: { 5 32 Num Lock }
[825] 93: { 6 64 Caps Lock }
[BX3] 94: { 7 128 Ins. }
[000] 95:
[BH1] 96: begin
[9H9] 97: reg.ah := 2; { Funktionsnummer }
[5F8] 98: intr(22, reg); { Ausführung }
[Y56] 99: ShiftStatus := reg.al;
[TU0] 100: end;
[000] 101:
[000] 102:
[000] 103:
[WYC] 104: function PrinterStatus(Kanal: byte): byte;
[000] 105:
[32D] 106: { ergibt den Status des Drucker-Kanals }
[MT8] 107: { Kanal: 1 = lpt1, 2 = lpt2, 3 = lpt3 }
[000] 108:
[GK6] 109: { Bit Bedeutung }
[9W4] 110: { ----- }
[048] 111: { 0 falsche Kanalnummer }
[T56] 112: { 1..2 (unbenutzt) }
[C75] 113: { 3 I/O-Fehler }
[149] 114: { 4 Drucker ist selektiert }
[YU6] 115: { 5 kein Papier mehr }
[518] 116: { 6 Drucker antwortet }
[DA7] 117: { 7 Drucker wartet }
[000] 118:
[BH1] 119: begin
[66E] 120: reg.dx := (Kanal and 3) - 1; { lpt1, lpt2 oder lpt3 }
[9H9] 121: reg.ah := 2; { Funktionsnummer }
[CF8] 122: intr(23, reg); { Ausführung }
[NN7] 123: PrinterStatus := reg.ah;
[TU0] 124: end;
[000] 125:
[000] 126:
[000] 127:
[FR8] 128: procedure SystemRestart;

```



```

[000] 129:
[SYH] 130: { Betriebssystem wird neu gebootet (wie [Ct
      rl-Alt-Del]) }
[000] 131:
[BH1] 132: begin
[EE3] 133:   intr(25, reg);
[TU0] 134: end;
[000] 135:
[000] 136:
[000] 137:
[TR9] 138: { ***** Bildschirm-Funktionen ***** }
[000] 139:
[000] 140:
[A21] 141: type
[F37] 142:   AdapterTyp = (mono, cga);
[000] 143:
[0FC] 144: { Adaptertyp-Voreinstellung ist CGA. }
[YHC] 145: { Bitte ändern, wenn nicht vorhanden }
[000] 146:
[VM1] 147: const
[WU7] 148:   adapter: AdapterTyp = cga;
[000] 149:
[000] 150:
[000] 151:
[DL7] 152: procedure PrintScreen;
[000] 153:
[TR7] 154: { Bildschirm-Hardcopy }
[000] 155:
[BH1] 156: begin
[B68] 157:   intr(5, reg); { Ausführung }
[TU0] 158: end;
[000] 159:
[000] 160:
[000] 161:
[92C] 162: procedure SetCursorSize(von, bis: byte);
[000] 163:
[ULK] 164: { Cursorgröße von Rasterzeile bis Rasterzei
      le setzen }
[9AH] 165: { CGA: 0 = oberste Rasterzeile, 7 = unters
      te Rasterzeile }
[54K] 166: { Mono: 0 = oberste Rasterzeile, 14 = unter
      ste Rasterzeile }
[000] 167:
[BH1] 168: begin
[BP9] 169:   reg.ah := 1; { Funktionsnummer }
[899] 170:   reg.ch := von; { Startzeile }
[GD8] 171:   reg.cl := bis; { Endezeile }
[ZV8] 172:   intr(16, reg); { Ausführung }
[TU0] 173: end;
[000] 174:
[000] 175:
[000] 176:
[DM6] 177: procedure HideCursor;
[000] 178:
[0X6] 179: { Cursor abschalten }
[000] 180:
[BH1] 181: begin
[VS5] 182:   SetCursorSize(32,32);
[TU0] 183: end;
[000] 184:
[000] 185:
[000] 186:
[DU6] 187: procedure ShowCursor;
[000] 188:
[G09] 189: { Strichcursor einschalten }
[000] 190:
[BH1] 191: begin
[705] 192:   case adapter of
[827] 193:     mono: SetCursorSize(12,14);
[RF6] 194:     cga: SetCursorSize(6,7);
[293] 195:   end; { case }
[TU0] 196: end;
[000] 197:
[000] 198:
[000] 199:
[OK7] 200: procedure BlockCursor;
[000] 201:
[1N8] 202: { Blockcursor einschalten }
[000] 203:
[BH1] 204: begin
[705] 205:   case adapter of
[AX6] 206:     mono: SetCursorSize(0,14);
[XD6] 207:     cga: SetCursorSize(0,7);
[293] 208:   end; { case }
[TU0] 209: end;
[000] 210:
[000] 211:
[000] 212:
[TV4] 213: procedure SetDisplayPage(nr: byte);
[000] 214:
[DMG] 215: { aktive Bildschirmseite im Textmodus festl
      egen }
[478] 216: { 40-Zeichen-Modus: nr = 0..7 }
[W68] 217: { 80-Zeichen-Modus: nr = 0..3 }
[000] 218:

```

```

[BH1] 219: begin
[KH9] 220:   reg.ah := 5; { Funktionsnummer }
[LU9] 221:   reg.bh := nr; { Bildschirmseite }
[VF8] 222:   intr(16, reg); { Ausführung }
[TU0] 223: end;
[000] 224:
[000] 225:
[000] 226:
[CVC] 227: procedure GetDisplayPage(var nr: byte);
[000] 228:
[5HD] 229: { aktive Bildschirmseite feststellen }
[OL9] 230: { nr = 0..7 für Textmodus bei CGA }
[X8F] 231: { nr = 0 für Grafikmodus oder Monochrom-Ada
      pter }
[000] 232:
[BH1] 233: begin
[UT9] 234:   reg.ah := 15; { Funktionsnummer }
[ZV8] 235:   intr(16, reg); { Ausführung }
[6T9] 236:   nr := reg.bh; { Bildschirmseite }
[TU0] 237: end;
[000] 238:
[000] 239:
[000] 240:
[Z7D] 241: function ScreenChar(Seite: byte): char;
[000] 242:
[PFK] 243: { Zeichen an Cursorposition aus Bildschirms
      eite lesen }
[XXF] 244: { Seite = 0..7 für Textmodus, Seite = 0 für
      Grafikmodus }
[000] 245:
[BH1] 246: begin
[DZ9] 247:   reg.ah := 8; { Funktionsnummer }
[DBB] 248:   reg.bh := Seite; { Bildschirmseite }
[LG8] 249:   intr(16, reg); { Ausführung }
[1X6] 250:   ScreenChar := chr(reg.al);
[TU0] 251: end;
[000] 252:
[000] 253:
[000] 254:
[CX9] 255: { ***** Disketten-Operationen ***** }
[000] 256:
[000] 257:
[A21] 258: type
[Z6A] 259:   SektorPuffer = array[0..511] of byte;
[000] 260:
[000] 261:
[000] 262:
[3X8] 263: procedure ResetDiskSystem;
[000] 264:
[37H] 265: { Diskettenlaufwerke und Controller zurücks
      etzen }
[000] 266:
[BH1] 267: begin
[8P9] 268:   reg.ah := 0; { Funktionsnummer }
[MW8] 269:   intr(19, reg); { Ausführung }
[TU0] 270: end;
[000] 271:
[000] 272:
[000] 273:
[2C8] 274: function DiskStatus: byte;
[000] 275:
[BZK] 276: { ergibt den Fehlerstatus der letzten Diske
      tten-Operation }
[000] 277:
[DZ7] 278: { Wert Bedeutung }
[N36] 279: { ----- }
[6D6] 280: { 0 kein Fehler }
[AC8] 281: { 1 ungültiger Befehl }
[ZVA] 282: { 2 Adreßmarke nicht gefunden }
[BLB] 283: { 3 Diskette ist schreibgeschützt }
[ZV8] 284: { 4 Sektor nicht gefunden }
[556] 285: { 8 DMA-Fehler }
[ET8] 286: { 16 Prüfsummenfehler }
[F4B] 287: { 32 Fehler im Disketten-Controller }
[SM8] 288: { 64 Spur nicht gefunden }
[5C9] 289: { 128 Laufwerk nicht bereit }
[000] 290:
[BH1] 291: begin
[6H9] 292:   reg.ah := 1; { Funktionsnummer }
[GG8] 293:   intr(19, reg); { Ausführung }
[SV5] 294:   DiskStatus := reg.ah;
[TU0] 295: end;
[000] 296:
[000] 297:
[000] 298:
[AGS] 299: procedure ReadSector(Laufwerk, Seite, Spur,
      Sektor: byte; var Puffer: SektorPuffer);
[000] 300:
[CSE] 301: { liest einen Diskettensektor in Puffer ein }
[000] 302:
[BH1] 303: begin
[9DA] 304:   reg.ah := 2; { Funktionsnummer }
[000] 305:   reg.dl := Laufwerk; { Laufwerksnummer
      0..1 }

```



```

[ZCZ] 306: reg.dh := Seite;      { Diskettenseite
0..1 }
[E49] 307: reg.ch := Spur;      { Spur 0..39 }
[99A] 308: reg.cl := Sektor;    { Sektor 1..9 }
[Z5E] 309: reg.bx := ofs(Puffer); { Offsetadresse P
uffer }
[XNE] 310: reg.es := seg(Puffer); { Segmentadresse
Puffer }
[GC9] 311: intr(19, reg);      { Ausführung }
[TU0] 312: end;
[000] 313:
[000] 314:
[000] 315:
[HE7] 316: procedure WriteSector(Laufwerk, Seite, Spur
, Sektor: byte; var Puffer: SektorPuffer);
[000] 317:
[XHC] 318: { schreibt Puffer in Diskettensektor }
[000] 319:
[BH1] 320: begin
[CDA] 321: reg.ah := 3;        { Funktionsnummer
}
[L1D] 322: reg.dl := Laufwerk;  { Laufwerksnummer
0..1 }
[ZCZ] 323: reg.dh := Seite;    { Diskettenseite
0..1 }
[E49] 324: reg.ch := Spur;      { Spur 0..39 }
[99A] 325: reg.cl := Sektor;    { Sektor 1..9 }
[Z5E] 326: reg.bx := ofs(Puffer); { Offsetadresse P
uffer }
[XNE] 327: reg.es := seg(Puffer); { Segmentadresse
Puffer }
[GC9] 328: intr(19, reg);      { Ausführung }
[TU0] 329: end;
[000] 330:
[000] 331:
[000] 332:
[XCK] 333: procedure VerifySector(Laufwerk, Seite, Spu
r, Sektor: byte);
[000] 334:
[4S9] 335: { überprüft Diskettensektor }
[000] 336:
[BH1] 337: begin
[FDA] 338: reg.ah := 4;        { Funktionsnummer
}
[L1D] 339: reg.dl := Laufwerk;  { Laufwerksnummer
0..1 }
[ZCZ] 340: reg.dh := Seite;    { Diskettenseite
0..1 }
[E49] 341: reg.ch := Spur;      { Spur 0..39 }
[99A] 342: reg.cl := Sektor;    { Sektor 1..9 }
[GC9] 343: intr(19, reg);      { Ausführung }
[TU0] 344: end;
[000] 345:
[000] 346:
[000] 347:
[U7T] 348: procedure FormatTrack(Laufwerk, Seite, Spur
, Sektor: byte; var Puffer: SektorPuffer);
[000] 349:
[829] 350: { formatiert Diskettenspur }
[000] 351:
[SHB] 352: { Puffer muß für jeden Sektor der }
[PHB] 353: { zu formatierenden Spur vier Bytes }
[8LA] 354: { an Informationen enthalten: }
[1Y6] 355: { 1. Spurnummer }
[V87] 356: { 2. Seitennummer }
[OB7] 357: { 3. Sektornummer }
[M7A] 358: { 4. Sektorgröße codiert als: }
[W16] 359: { 0: 128 Bytes }
[626] 360: { 1: 256 Bytes }
[Z16] 361: { 2: 512 Bytes }
[Z36] 362: { 3: 1024 Bytes }
[000] 363:
[BH1] 364: begin
[KDA] 365: reg.ah := 5;        { Funktionsnummer
}
[L1D] 366: reg.dl := Laufwerk;  { Laufwerksnummer
0..1 }
[ZCZ] 367: reg.dh := Seite;    { Diskettenseite
0..1 }
[E49] 368: reg.ch := Spur;      { Spur 0..39 }
[GC9] 369: intr(19, reg);      { Ausführung }
[Z5E] 370: reg.bx := ofs(Puffer); { Offsetadresse P
uffer }
[XNE] 371: reg.es := seg(Puffer); { Segmentadresse
Puffer }
[TU0] 372: end;
[000] 373:
[000] 374:
[000] 375:
[7T6] 376: { ***** RS-232C-Routinen ***** }
[000] 377:
[000] 378:
[GRM] 379: procedure InitRS232(Kanal, Baud, Parity, St
opBits, DataBits: byte);

```

```

[000] 380:
[R8G] 381: { Parameter für serielle Schnittstelle setz
en }
[000] 382:
[LM9] 383: { Kanal: 0 = com0, 1 = com1
}
[3X8] 384: { Baud: 0 = 110, 1 = 150, 2 = 300, 3 =
600,
}
[BZ7] 385: { 4 = 1200, 5 = 2400, 6 = 4800, 7
= 9600 }
[N9C] 386: { Parity: 0 = keine, 1 = ungerade (odd),
}
[YRA] 387: { 2 = keine, 3 = gerade (even)
}
[Y2C] 388: { StopBits: 1 = 1 Stop-Bit, 2 = 2 Stop-Bits
}
[VAD] 389: { DataBits: 7 = 7 Daten-Bits, 8 = 8 Daten-B
its
}
[000] 390:
[BH1] 391: begin
[F26] 392: reg.dx := Kanal and 1;
[136] 393: reg.al := (Baud and 7);
[14B] 394: reg.al := (reg.al shl 2) or (Parity and 3
);
[PZC] 395: reg.al := (reg.al shl 2) or (StopBits and
3) or 1;
[OMC] 396: reg.al := (reg.al shl 1) or ((DataBits -
7) and 1);
[8P9] 397: reg.ah := 0;        { Funktionsnummer }
[VU8] 398: intr(20, reg);      { Ausführung }
[TU0] 399: end;
[000] 400:
[000] 401:
[000] 402:
[HSC] 403: procedure SendRS232(Kanal: byte; ch: char);
[000] 404:
[CBG] 405: { Zeichen ch über serielle Schnittstelle se
nden }
[000] 406:
[BH1] 407: begin
[TTB] 408: reg.dx := Kanal and 1; { com0 oder com
1 }
[GKD] 409: reg.al := ord(ch);   { zu sendendes Z
eichen }
[BKA] 410: reg.ah := 1;        { Funktionsnumme
r }
[VP9] 411: intr(20, reg);      { Ausführung }
[TU0] 412: end;
[000] 413:
[000] 414:
[000] 415:
[SNE] 416: procedure ReadRS232(Kanal: byte; var ch: ch
ar);
[000] 417:
[5YG] 418: { Zeichen ch von serieller Schnittstelle ei
nlesen }
[000] 419:
[BH1] 420: begin
[K4B] 421: reg.dx := Kanal and 1; { com0 oder com1
}
[9DA] 422: reg.ah := 2;        { Funktionsnummer
}
[RA9] 423: intr(20, reg);      { Ausführung }
[LRC] 424: ch := chr(reg.al);   { empfangenes Zei
chen }
[TU0] 425: end;
[000] 426:
[000] 427:
[000] 428:
[KSC] 429: function RS232Status(Kanal: byte): integer;
[000] 430:
[1KF] 431: { Status der letzten seriellen Operation ho
len }
[000] 432:
[729] 433: { Bit Modemstatus
}
[YW7] 434: { -----
--
}
[V69] 435: { 0 DCTS (Delta Clear To Send)
}
[6F9] 436: { 1 DDSR (Delta Data Set Ready)
}
[MOC] 437: { 2 TREI (Trailing Edge Ring Indicator
)
}
[17B] 438: { 3 DDCD (Delta Data Carrier Detect)
}
[EH8] 439: { 4 CTS (Clear To Send)
}
[019] 440: { 5 DSR (Data Set Ready)
}
[3X8] 441: { 6 RI (Ring Indicator)
}
[HL9] 442: { 7 DCD (Data Carrier Detect)
}

```



```

[YW7] 443: { -----
[N3A] 444: { Bit Leitungsstatus
}
[YW7] 445: { -----
[S38] 446: { 8 Daten bereit
}
[G69] 447: { 9 Überlauffehler
}
[LG9] 448: { 10 Paritätsfehler
}
[ET8] 449: { 11 Framing-Fehler
}
[ZV8] 450: { 12 Unterbrechung
}
[18D] 451: { 13 Übertragungs-Halteregister leer
}
[HED] 452: { 14 Übertragungs-Schieberegister leer
}
[CRB] 453: { 15 Gegenstation nicht bereit
}
[000] 454:
[BH1] 455: begin
[K4B] 456: reg.dx := Kanal and 1; { com0 oder com1
}
[CDA] 457: reg.ah := 3; { Funktionsnummer
}
[RA9] 458: intr(20, reg); { Ausführung }
[HKA] 459: RS232Status := reg.ax; { Statuswort }
[TU0] 460: end;
[000] 461:
[000] 462:
[000] 463:
[9TA] 464: { ***** Zeit- und Datumsfunktionen ***** }
[000] 465:
[XRH] 466: { Achtung, diese Routinen funktionieren nur
beim AT }
[MXG] 467: { sowie bei einigen kompatiblen einfachen P
Cs mit }
[59F] 468: { Echtzeituhr (unter anderem beim Schneider
PC) }
[000] 469:
[000] 470:
[RVA] 471: function toByte(b: byte): byte;
[000] 472:
[999] 473: { BCD in Binärformat wandeln }
[000] 474:
[BH1] 475: begin
[KC8] 476: toByte := (b shr 4) * 10 + (b and 15);
[TU0] 477: end;
[000] 478:
[000] 479:
[000] 480:
[409] 481: function toBCD(b: byte): byte;
[000] 482:
[8E9] 483: { Binärformat in BCD wandeln }
[000] 484:
[BH1] 485: begin
[FK8] 486: toBCD := ((b div 10) shl 4) + (b mod 10);
[TU0] 487: end;
[000] 488:
[000] 489:
[000] 490:
[4VE] 491: procedure GetTime(var Stunde, Minute, Sekun
de: byte);
[000] 492:
[CH8] 493: { aktuelle Uhrzeit lesen }
[000] 494:
[BH1] 495: begin
[C89] 496: reg.ah := 2; { Funktionsnummer }
[NY7] 497: intr(26, reg); { Ausführung }
[7M7] 498: Stunde := toByte(reg.ch);
[YP7] 499: Minute := toByte(reg.cl);
[EV7] 500: Sekunde := toByte(reg.dh);
[TU0] 501: end;
[000] 502:
[000] 503:
[000] 504:
[0EG] 505: procedure SetTime(Stunde, Minute, Sekunde:
byte);
[000] 506:
[CZ8] 507: { aktuelle Uhrzeit setzen }
[000] 508:
[BH1] 509: begin
[RRA] 510: reg.ch := toBCD(Stunde); { Stunde 0..23
}
[6SA] 511: reg.cl := toBCD(Minute); { Minute 0..59
}
[RKB] 512: reg.dh := toBCD(Sekunde); { Sekunde 0..5
9 }
[XPH] 513: reg.dl := 1; { automatische Sommerzeitums
chaltung ein }
[HP9] 514: reg.ah := 3; { Funktionsnummer }

```

```

[5W8] 515: intr(26, reg); { Ausführung }
[TU0] 516: end;
[000] 517:
[000] 518:
[000] 519:
[7VD] 520: procedure GetDate(var Tag, Monat, Jahr: byt
e);
[000] 521:
[568] 522: { aktuelles Datum lesen }
[000] 523:
[BH1] 524: begin
[FH9] 525: reg.ah := 4; { Funktionsnummer }
[LG8] 526: intr(26, reg); { Ausführung }
[YH5] 527: Tag := toByte(reg.dl);
[M37] 528: Monat := toByte(reg.dh);
[1B6] 529: Jahr := toByte(reg.cl);
[TU0] 530: end;
[000] 531:
[000] 532:
[000] 533:
[BVB] 534: procedure SetDate(Tag, Monat, Jahr: byte);
[000] 535:
[VD8] 536: { aktuelles Datum setzen }
[000] 537:
[BH1] 538: begin
[949] 539: reg.dl := toBCD(Tag); { Tag 1..31 }
[Y6A] 540: reg.dh := toBCD(Monat); { Monat 1..12 }
[TP9] 541: reg.cl := toBCD(Jahr); { Jahr 00..99 }
[N89] 542: reg.ah := 5; { Funktionsnummer }
[NY7] 543: intr(26, reg); { Ausführung }
[TU0] 544: end;
[000] 545:
[84A] 546: { ----- Ende BIOS.INC
----- }
[000] 547:
[000] 548:

```

Listing 4. Alle nützlichen BIOS-Routinen für Turbo Pascal

```

[9K4] 0: PROGRAM DruckOK;
[000] 1:
[XHK] 2: { Register nach dem Turbo Pascal Handbuch d
eklarieren }
[D8F] 3: VAR Regs : RECORD ax,bx,cx,dx,bp,si,di,
ds,es,fi: INTEGER;
[000] 4: END;
[000] 5:
[B21] 6: BEGIN
[XHE] 7: { Druckerstatus lesen ist Unterfunktion 2
}
[000] 8: Regs.ax := $0200;
[Y6B] 9: { Druckernummer in DX: 0 = LPT1, 1 = LPT2
}
[000] 10: Regs.dl := 0;
[000] 11: { BIOS Funktion aufrufen }
[000] 12: Intr (23, Regs);
[000] 13: { Status kommt im AH-Register zurück
}
[000] 14: { Drittes Bit "E/A-Fehler" ist maßgebend
}
[000] 15: IF (Regs.ax AND $800) <> 0 THEN
[000] 16: Halt (1)
[000] 17: ELSE
[000] 18: Halt (0);
[000] 19: END.

```

Listing 5. Ein kleines Turbo-Pascal-Programm überprüft den Druckerstatus

```

[3U1] 0: ECHO OFF
[HY2] 1: druckok
[7V7] 2: IF ERRORLEVEL 1 GOTO :notready
[AD5] 3: print %1 %2 %3 %4 %5 %6 %7 %8 %9
[022] 4: GOTO :end
[BZ2] 5: :notready
[X1D] 6: ECHO Der Drucker ist nicht betriebsbereit!
[Y0] 7: :end

```

Listing 6. Diese Stapeldatei verhindert Fehler, wenn der Drucker nicht eingeschaltet ist



Der PC (Teil 3) - glasklar

Bisher lag der Schwerpunkt unseres Kurses auf den Grundlagen der PC-Programmierung und des BIOS. In diesem Teil gehen wir einen Schritt »nach oben« zu MS-DOS, dem Betriebssystem des PC. Mit den MS-DOS-Funktionen lassen sich Dateien, Verzeichnisse (Directories) und Prozesse bequem und einfach erzeugen, löschen und manipulieren – Dinge, die jeder braucht, wenn er mit dem PC arbeitet.

Alle MS-DOS-Funktionen werden über einen einzigen speziell dafür reservierten Interrupt, nämlich den Interrupt 21hex, aufgerufen. Welche der vielen Funktionen ausgeführt wird, das bestimmt die Funktionsnummer, die im Prozessor-Register AH übergeben wird. Beispielsweise sieht ein Aufruf der MS-DOS-Funktion Nummer 0 (Programm beenden) in Assembler folgendermaßen aus:

```
mov ah,0 ; MS-DOS-Funktion 0
int 21H ; DOS-Interrupt ausführen
```

Viele MS-DOS-Funktionen benötigen noch zusätzliche Parameter. Diese werden im Bedarfsfall in weiteren Prozessor-Registern übergeben. Außerdem sind die meisten Betriebssystem-Routinen in der Lage, Fehler zu erkennen und eine Fehlernummer (ebenfalls in einem Prozessor-Register) zurückzugeben.

Um die verschiedenen Funktionen, die sich zum Teil überschneiden, im richtigen Zusammenhang zu verstehen, müssen wir kurz die Geschichte von MS-DOS erzählen. Die Entwicklung von MS-DOS erfolgte in bisher

drei großen Schritten: Die ersten Versionen (mit Versionsnummern 1.x) des MS-DOS-Betriebssystems lehnten sich aus Gründen der damals notwendig erscheinenden Kompatibilität zu CP/M sehr stark an dieses Betriebssystem an. Erst die MS-DOS Version 2 brachte die heute allgemein verwendeten Unix-ähnlichen Funktionen zur Steuerung des Dateisystems und eine konsequente, einheitliche Fehlerbehandlung. Aus diesem Grund werden die alten Versionen 1.x kaum mehr verwendet.

Zusätzlich zum bekannten Dateisystem arbeiten fast alle ab Version 2 neu hinzugekommenen Funktionsaufrufe mit der Vereinbarung, daß ein aufgetretener Fehler bei Rückkehr der Routine durch ein gesetztes Carry-Flag signalisiert wird. Diese Vereinbarung macht die Erkennung eines Fehlers sehr einfach.

Ab der MS-DOS-Version 3 ist neben einigen weiteren Funktionen, die unter anderem Netzwerke unterstützen, eine wichtige Funktion zur Fehlerbehandlung neu hinzugekommen. Es

handelt sich dabei um die Funktion 59hex »Get Extended Error«. Diese Funktion liefert bei Aufruf genauere Angaben über Art und Ursprung eines Fehlers bei der jeweils zuletzt aufgerufenen MS-DOS-Funktion. Außerdem gibt sie Hinweise darüber, welche Aktionen seitens des laufenden Programms zur Behandlung des Fehlers sinnvoll sind.

Insgesamt erhalten Sie vier Angaben über den Fehler: erweiterter Fehlercode (Extended Error Code), Fehlerklasse (Error Class), Aktionsvorschlag (Suggested Action) und Fehlerort (Location). Die Bedeutung des erweiterten Fehlercodes läßt sich anhand der Tabelle 1 feststellen. Nur die ersten zwölf Fehlercodes in dieser Tabelle werden von den MS-DOS-Funktionen direkt zurückgegeben, alle weiteren Codes erscheinen nur durch die Funktion 59hex als »erweiterte Fehlercodes«. Die möglichen Werte für Fehlerklasse, Aktionsvorschlag und Fehlerort faßt Tabelle 2 zusammen. Diese Werte werden ebenfalls nur durch die Funktion 59hex zur Verfügung gestellt.

Die Anzahl der MS-DOS-Funktionsaufrufe wurde von DOS-Version zu DOS-Version erhöht; insgesamt sind die Funktionsnummern 0 bis 127 reserviert. Wie bereits erwähnt, laufen alle diese Funktionsaufrufe über den Interrupt 21hex. Daneben verwendet MS-DOS noch weitere Interrupts, wie bereits im Teil 2 dieses Kurses beschrieben. Bis auf den Funktions-Interrupt 21hex sind die DOS-Interrupts in erster Linie für die interne Verwendung durch MS-DOS selbst vorgesehen. Microsoft und IBM raten daher dringend von der Verwendung anderer DOS-Interrupts als 21hex ab. Einige dieser Interrupts wurden mittlerweile durch gleichartige oder verbesserte Funktionsaufrufe ergänzt und sind nur noch aus Kompatibilitätserwägungen in den neuen Versionen enthalten. Es gibt keine Garantie, daß MS-DOS derartige DOS-Interrupts auch in zukünftigen Versionen unterstützt.

Die sehr umfangreiche Referenztafel am Ende dieses Artikels gibt einen Überblick über alle MS-DOS-Funktionsaufrufe mit Angabe der MS-DOS-Version, ab der die jeweiligen Funktionen verfügbar wurden. Eine ganze Reihe davon stammen noch aus der Anfangszeit des MS-DOS, wo die Kompatibilität zum CP/M-Betriebssystem noch im Vordergrund stand. Sowohl IBM als auch Microsoft raten allerdings von ihrer Verwendung bis auf wenige Ausnahmen ab. Statt dessen sind neue, Unix-ähnliche Funktionsaufrufe vorgesehen, die die alten CP/M-kompatiblen Aufrufe ersetzen. In Tabelle 3 finden Sie eine Gegenüberstellung dieser alten

Funktionen und der neuen, die vorzugsweise benutzt werden sollen. Grundsätzlich gilt: Wenn es für eine bestimmte Aufgabe (beispielsweise das Öffnen einer Datei) mehrere Funktionen gibt, dann verwenden Sie immer die neuere, also diejenige mit der höheren Funktionsnummer. Wenn es allerdings wichtig ist, daß Ihr Programm auch mit älteren MS-DOS-Versionen wie etwa 2.x arbeitet, dann dürfen Sie umgekehrt keine MS-DOS-3-Routinen benutzen.

MS-DOS-Funktionen können auch direkt aus Hochsprachen aufgerufen werden

Der eigentliche Sinn und Zweck der MS-DOS-Routinen ist es, den einzelnen Programmen einen hinreichend komfortablen Rahmen zur Verfügung zu stellen, um die einzelnen Komponenten des PC beziehungsweise des Betriebssystems anzusprechen. Normalerweise erfolgt der Aufruf der Funktionen daher aus einem Assemblerprogramm heraus, direkt aus der Maschinenebene also. Aber auch mit einer Hochsprache wie beispielsweise Turbo Pascal läßt sich auf einem MS-DOS-Computer die ganze Vielzahl von Betriebssystemfunktionen ansprechen. Wohlgedenkt, auch Turbo Pascal benutzt natürlich MS-DOS-Funktionsaufrufe, um Aufgaben wie Dateiverwaltung oder Ein-/Ausgabeoperationen zur erfüllen. Doch der Aufruf dieser MS-DOS-Funktionen ist direkt in den Compiler »eingebaut« und wird automatisch durchgeführt. Was uns jedoch an dieser Stelle interessiert, ist eine Methode zum direkten, gesteuerten Aufruf von Betriebssystem-Funktionen. Mit solch einer Methode kann man sich auch Betriebssystem-Routinen zunutze machen, die Turbo Pascal standardmäßig nicht unterstützt. Bei Turbo Pascal 4.0 sind das allerdings vergleichsweise wenige Routinen. Bei Turbo Pascal 3.0 gehören beispielsweise die Speicherverwaltung oder die Datums- und Zeitfunktionen dazu. Speziell bei der Speicherverwaltung ist allerdings Vorsicht geboten. MS-DOS arbeitet mit sogenannten Speicherblöcken und kennt drei Funktionen, um Speicher zu belegen, freizugeben oder die Größe eines Speicherblocks zu verändern (MS-DOS-Funktionen 72 bis 74). Bei der Verwendung von Hochsprachen wie beispielsweise Turbo Pascal sollte man es nach Möglichkeit vermeiden, mit diesen Funktionen zu arbeiten. Turbo Pascal besitzt, wie andere Compiler auch, eine eigene, eingebaute Speicherverwaltung, die durcheinandergeraten könnte, wenn man die Speicherverwaltungsfunk-

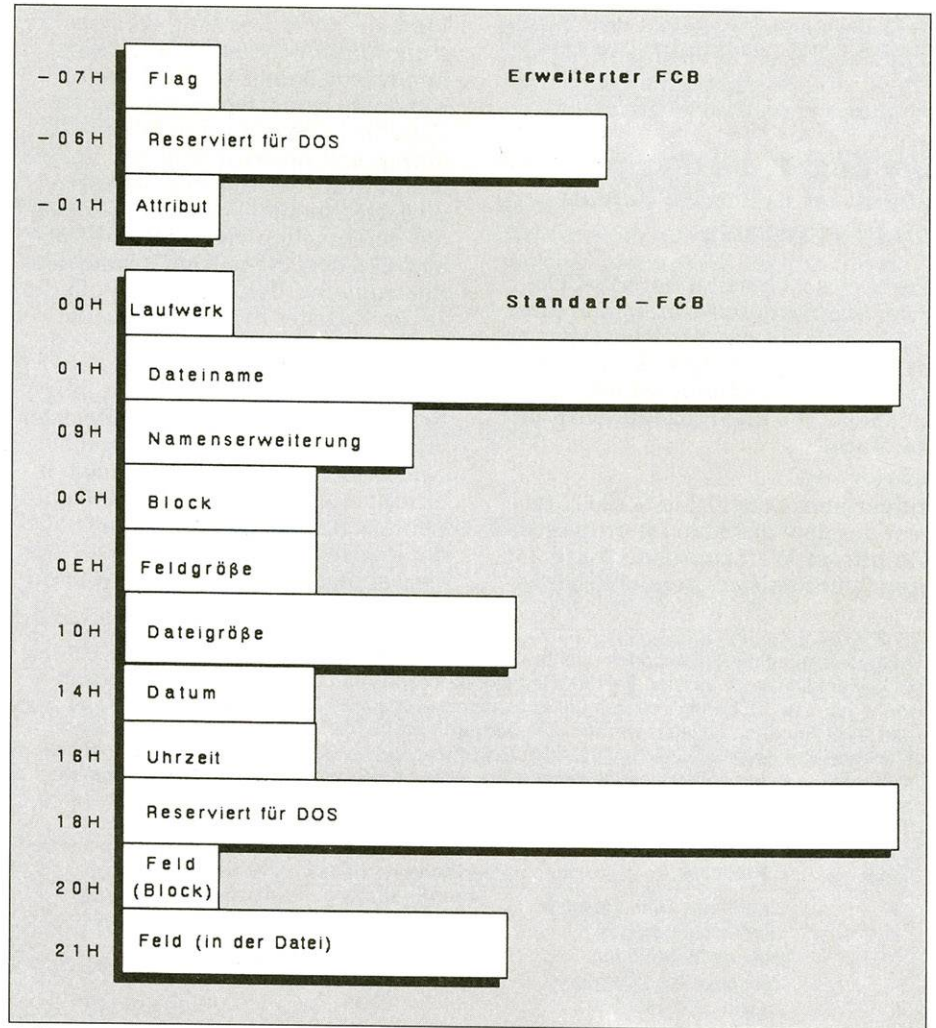


Bild 1. Der Aufbau eines File-Control-Blocks (FCB)

tionen des Betriebssystems ohne genaue Überlegung benutzt. Generell sollten Sie auf die MS-DOS-Funktionen verzichten, sofern die benutzte Hochsprache äquivalente Funktionen bietet. Dies betrifft neben der Speicherverwaltung vor allem auch den Datei- und den Ein-/Ausgabebereich.

Gleichwohl lohnt es sich in vielen Fällen, einmal näher auf die verschiedenen MS-DOS-Funktionen einzugehen. Allerdings lassen sich diese

Funktionen normalerweise nur mit einem soliden Fundament an Assembler-Programmierkenntnissen nutzen. Da sind Prozessor-Register mit bestimmten Werten zu laden, da ist das Flag-Register zu beachten und was der Dinge mehr sind. Wer sich nicht mit Assembler auskennt und vor allem kein MS-DOS Reference Manual zur Hand hat, dem war es bisher verwehrt, die große Vielfalt der MS-DOS-Funktionen in seinen

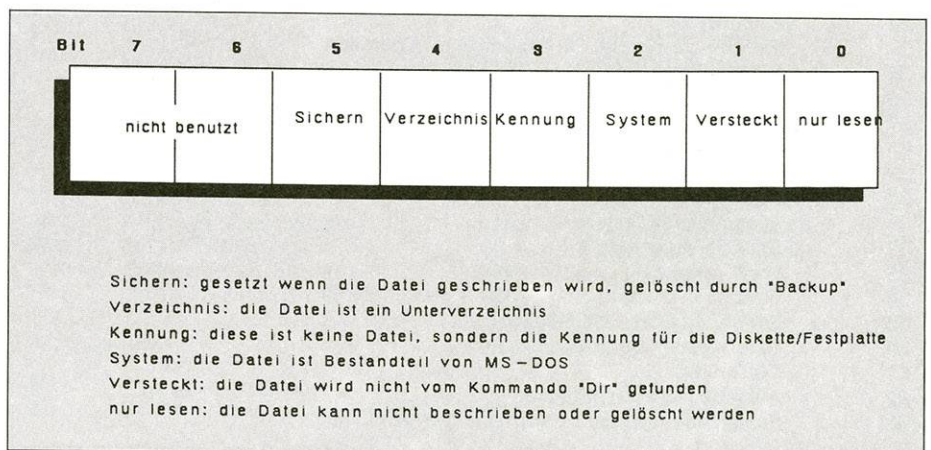


Bild 2. Dateiattribute unter MS-DOS

Programmen einzusetzen. Wir zeigen daher an dieser Stelle einen Weg auf, alle MS-DOS-Funktionen bequem unter Turbo Pascal 3.0 aufzurufen.

Der Zugriff auf die Prozessor-ebene ist in Turbo Pascal nicht schwierig

Turbo Pascal bietet unter MS-DOS zwei Standardprozeduren zum direkten Zugriff auf das Betriebssystem an. Es handelt sich dabei um die sehr allgemeinen Prozeduren »intr« und »msdos«. Die Intr-Prozedur wird in der Form

intr(nr, reg)

aufgerufen. Dabei ist »nr« die Nummer des gewünschten Interrupts, also ein Integer-Wert zwischen 0 und 255. »reg« (für Register) ist eine Record-

Variable zur Darstellung der Registerstruktur des 8088/8086-Prozessors und wurde bereits in der letzten Kursfolge beschrieben. Die Register-Variable repräsentiert praktisch die einzelnen Prozessor-Register. Vor der Ausführung des Interrupts werden die tatsächlichen Prozessor-Register mit den Inhalten der Register-Variablen geladen. Nach Ausführung des Interrupts werden umgekehrt die Prozessor-Register in den Register-Record gerettet.

Auf diese Art und Weise ist es möglich, den Interrupt-Routinen Registerinhalte als Parameter zu übergeben. Andererseits kann man nach dem Interrupt-Aufruf die von der Interrupt-Routine zurückgegebenen Werte aus der Register-Variablen auslesen. Wir verwenden allerdings eine etwas ver-

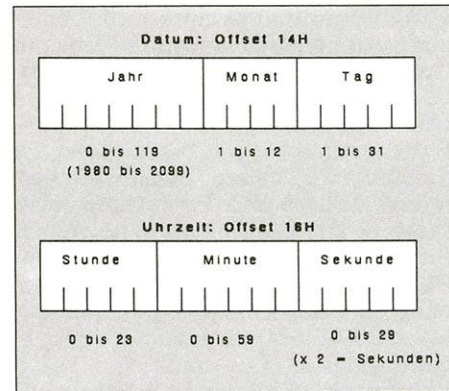


Bild 3. Aufbau von Dateidatum und -Zeit

feinerte Definition für diesen Registertyp. Der 8088/8086-Prozessor kann ja die Register AX, BX, CX und DX wahlweise auch als ein Paar aus zwei 8-Bit-Registern ansprechen. Um sich bei Verwendung von 8-Bit-Werten umständliche Umrechnungen zu ersparen, ist der Register-Record in der MS-DOS-Bibliothek als vollständig varianter Record (»Union«) realisiert. So kann man beispielsweise das AX-Register ganz normal mit »reg.ax« als 16-Bit-Register ansprechen, gleichzeitig aber auch als Paar aus den zwei 8-Bit-Registern »reg.al« und »reg.ah«.

Auch die Standardprozedur »msdos« von Turbo Pascal arbeitet mit der eben beschriebenen Registervariablen. Der Aufruf geschieht in der Form:

msdos(reg)

Der einzige Unterschied zur Intr-Prozedur ist also, daß keine Interrupt-Nummer angegeben wird. Die MS-DOS-Prozedur ruft immer den Funktions-Interrupt 21hex des MS-DOS-Systems auf. Über diesen Interrupt werden, wie bereits beschrieben, alle Funktionsaufrufe abgewickelt. Dazu wird im Register AH die Nummer der gewünschten Funktion übergeben. In Turbo Pascal sieht daher der Aufruf beispielsweise der DOS-Funktion 0 (Programm beenden) so aus:

```
reg.ah := 0;
msdos(reg);
```

Das Ergebnis dieses Funktionsaufrufes ist die Beendigung des gerade laufenden Turbo-Programms. Der Aufruf entspricht also der Turbo-Prozedur »halt«.

In der Include-Datei »msdos.inc« (Listing 1) wird eine vereinfachte Prozedur »dos« zum Aufruf der MS-DOS-Funktionen verwendet. Außerdem wird von der Möglichkeit der genauen Fehlerfeststellung durch MS-DOS Gebrauch gemacht. Die Turbo-Prozedur »GetXErrCode« ruft die Funktion 59hex auf und liefert die Ergebnisse an die vier globalen Variablen »ErrorCode«, »ErrorClass«,

Aufgelistet sind die Fehlercodes und ihre Bedeutung, wie sie von den meisten neuen MS-DOS-Funktionen ab Version 2 im AX-Register zurückgeliefert werden. Bei den Fehlernummern von 1 bis 12 handelt es sich um sogenannte »einfache Fehlercodes«, die direkt von den DOS-Routinen geliefert werden. Die Codes 13 bis 35, die »erweiterten Fehlercodes«, stammen von der MS-DOS-Funktion 59H (Get Extended Error). Die Fehlernummern 36 bis 49 sind für zukünftige Verwendung reserviert; Fehlermeldungen zwischen 50 und 79 sowie ab 84 sind bei Netzwerk-Betrieb vorgesehen.

Code	Bedeutung
1	Ungültiger Funktionscode
2	Datei nicht gefunden
3	Pfad nicht gefunden
4	Zu viele offene Dateien
5	Zugriff nicht möglich
6	Ungültige Dateinummer
7	Speicherkontrollblock zerstört
8	Unzureichende Speicherkapazität
9	Ungültige Speicherkontrollblockadresse
10	Ungültige Umgebung
11	Ungültiges Format
12	Ungültiger Zugriffscode
13	Ungültige Daten
14	(Reserviert)
15	Ungültiges Laufwerk
16	Versuch, aktuelles Verzeichnis zu löschen
17	Nicht das gleiche Gerät
18	Keine weiteren Dateien
19	Diskette hat Schreibschutz
20	Fehlerhafte Disketten-Einheit
21	Laufwerk nicht bereit
22	Unzulässiges Disketten-Kommando
23	CRC-Fehler
24	Falsche Längenangabe bei Disketten-Operation
25	Suchfehler
26	Keine MS-DOS-Diskette
27	Sektor nicht gefunden
28	Drucker ohne Papier
29	Schreibfehler
30	Lesefehler
31	Allgemeiner Fehler
32	Sharing-Verletzung
33	Lock-Verletzung
34	Falsche Diskette
35	Zugriff auf FCB nicht möglich
80	Datei existiert bereits
81	(Reserviert)
82	Aufgabe nicht durchführbar
83	Fehler innerhalb der Fehlerbehandlungsroutine

Tabelle 1. Die Fehlermeldungen der MS-DOS-Funktionen

»ErrorAction« und »ErrorLocation«. Der Aufruf von »GetXErrCode« erfolgt automatisch nach jedem erkannten MS-DOS-Fehler über die allgemeine Prozedur »Error«. Die vier zusätzlichen Fehlervariablen können vom Turbo-Anwenderprogramm ebenso wie die Variable »ErrorNum« (enthält den einfachen Fehlercode) ausgewertet werden. Nach einer Abfrage sollten diese Variablen mittels der Prozedur »ClearError« wieder auf Null zurückgesetzt werden. Erwähnenswert ist noch die Stringvariable »ErrorMessage«, die den Text der letzten Fehlermeldung enthält, ebenso wie die Schaltervariable »trap«, eine Variable vom Typ Boolean, die die Ausgabe von Fehlermeldungen regelt. Hat »trap« den Wert »true«, so setzt die Fehleroutine nur die entsprechenden Fehlervariablen, der Programmablauf wird jedoch nicht unterbrochen. Ist »trap« dagegen auf »false« gesetzt, dann erfolgt bei jedem Fehler eine Meldung am Bildschirm, und das Programm bricht ab. Bei früheren MS-/PC-DOS-Versionen als 3.0 ist der Funktionsaufruf 59hex, und damit die Durchführung der Prozedur »GetXErrCode«, nicht möglich. In diesem Falle sollte diese Prozedur aus der MS-DOS-Bibliothek entfernt werden.

Die MS-DOS-Bibliothek enthält die folgenden Module:

(1) Globale Vereinbarungen: Hier werden die für die Fehlerbehandlung wichtigen Variablen als typisierte Konstanten definiert. Außerdem erfolgt die Typdefinition und Variablenvereinbarung für einfache Zeiger auf Speicherbytes für Pfadnamen, Dateinummern und schließlich für die Registerstruktur des 8086-Prozessors. Diese letzte Vereinbarung ist die wichtigste Grundlage für alle Bibliothek-Module rund um das MS-DOS-Betriebssystem. Die Prozessor-Register sind als varianter Record definiert, so daß sowohl 16- als auch 8-Bit-Register direkt und einfach mit ihrem Namen angesprochen werden können.

(2) DOS-Funktionen aufrufen: Dieses Modul besteht aus den beiden Prozeduren »dos« und »carry«. »dos« wird mit der gewünschten DOS-Funktionsnummer als Parameter aufgerufen und setzt einen global definierten Register-Record »reg« voraus. Die Funktion »carry« testet den Status des Carry-Flags innerhalb des Flag-Wortes des Register-Records. Wenn »carry« den Wert »true« liefert, war das Carry-Flag gesetzt, was bei den meisten DOS-Funktionen gleichbedeutend mit dem Auftreten eines Fehlers ist.

(3) Fehlerbehandlung: Dieses Modul besteht aus den beiden schon besprochenen Prozeduren »GetXErrCode«

und »Error«. Falls eine MS-/PC-DOS-Version kleiner als 3.0 verwendet wird, ist die Prozedur »GetXErrCode« zu löschen, ebenso der Aufruf dieser Prozedur innerhalb der Error-Prozedur.

Auf die Realisierung spezieller Funktionen in Form von Pascal-Prozeduren wurde verzichtet. Die Anzahl der nützlichen MS-/PC-DOS-Funktions-

aufrufe ist so hoch, daß eine solche Bibliothek den Rahmen dieses Kurses mit Leichtigkeit sprengen würde. Die Angaben aus der Referenztafel ermöglichen es aber jedem Anwender von Turbo Pascal, sich die benötigten Prozeduren selbst zusammenzustellen.

Drei Gruppen der MS-DOS-Funktionen sind besonders wichtig, da sie

Wert im Register BH: Fehlerklasse	
1	Unzureichende Ressourcen wie Speicher oder Kanäle
2	Kein Fehler, sondern ein nur zeitweiliges Problem
3	Nicht autorisierter Dateizugriff
4	Interner Fehler im Betriebssystem
5	Hardware-Fehler
6	Nicht vom aktuellen Prozeß verursachter DOS-Fehler
7	Fehler im Anwender-Programm
8	Datei oder anderes Betriebsmittel wurde nicht gefunden
9	Datei oder Betriebsmittel in ungültigem Format oder sonstwie unbrauchbar
10	Datei oder Betriebsmittel für Zugriff gesperrt
11	Falsche oder schadhafte Diskette oder anderes Problem mit einem Speichermedium
12	Sonstiger Fehler
Wert im Register BL: vorgeschlagene Aktion	
1	Gewünschte Aktion nochmals versuchen, bei Fehlschlag Benutzer benachrichtigen
2	Gewünschte Aktion nach einer Pause nochmals versuchen
3	Wenn der Benutzer eine Eingabe gemacht hat (wie Laufwerks- oder Dateiname), dann nochmals um (korrekte) Eingabe bitten
4	Programm geordnet beenden
5	Programm sofort ohne weitere Verzögerungen abbrechen
6	Informeller Fehler, kann ignoriert werden
7	Den Benutzer zu bestimmten Tätigkeiten auffordern (Diskettenwechsel etc.)
Wert im Register CH: Ort der Fehlerursache	
1	Unbekannt
2	Fehler liegt bei einem blockorientierten Gerät (beispielsweise Diskettenstation)
3	Fehler liegt in einem angeschlossenen Netzwerk
4	Fehler liegt bei einem zeichenweise anzusprechenden Gerät (beispielsweise Drucker)
5	Fehler liegt im Schreib-/Lesespeicher (RAM)

Tabelle 2. Fehlerklasse, Fehlerort und vorgeschlagene Reaktion, wie sie von der MS-DOS-Funktion 59H (Get Extended Error) geliefert werden

Alte Funktion		ersetzt durch	
00H	Programm beenden	4CH	Prozeß beenden
0FH	Datei eröffnen	3DH	Handle eröffnen
10H	Datei erschließen	3EH	Handle erschließen
11H	Ersten Eintrag suchen	4EH	Erste Datei finden
12H	Nächster Eintrag suchen	4FH	Nächste Datei finden
13H	Datei löschen	41H	Verzeichniseintrag löschen
14H	serielles Lesen	3FH	Handle lesen
15H	serielles Schreiben	3DH	Handle eröffnen
16H	Datei erzeugen	3CH	Handle erzeugen
		5AH	temporäre Datei erzeugen
		5BH	neue Datei erzeugen
17H	Datei umbenennen	56H	Verzeichniseintrag ändern
21H	direktes Lesen	3FH	Handle lesen
22H	direktes Schreiben	40H	Handle schreiben
23H	Dateigröße lesen	42H	Dateizeiger verschieben
24H	Feld in der Datei setzen	42H	Dateizeiger verschieben
26H	neues PSP erzeugen	4CH	Prozeß laden und starten
27H	direktes Lesen (Block)	3FH	Handle lesen
28H	direktes Schreiben (Block)	40H	Handle schreiben
Alter Interrupt		ersetzt durch	
20H	Programm beenden	4CH	Prozeß beenden
27H	Beenden, aber im Speicher bleiben	31H	Prozeß behalten

Tabelle 3. Alte MS-DOS-Funktionen und deren Ersatz

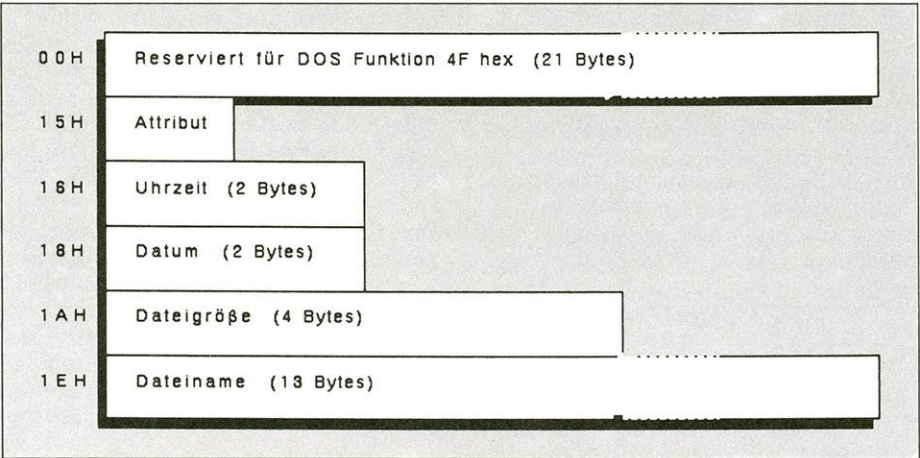


Bild 4. Datenstruktur der Directory-Funktionen

häufig gebraucht werden und von anderen Quellen (zum Beispiel vom BIOS) nicht angeboten werden. Diese Gruppen sind die Dateifunktionen, die Verzeichnisfunktionen und die Prozeßfunktionen. Nun wollen wir jede Gruppe kurz betrachten und gewisse Einzelheiten der verschiedenen Funktionen erklären.

Die Dateifunktionen von MS-DOS sind der Schlüssel zum Massenspeicher

In Tabelle 4 sehen Sie die Dateifunktionen von MS-DOS aufgelistet. Hier ist zu bemerken, daß Funktionen mit Nummern kleiner als 30hex alte Funktionen sind, die durch neuere ersetzt wurden. Jedoch stehen diese alten Funktionen noch zur Verfügung. Manche Programmierer setzen immer noch die alten Funktionen ein, weil diese unter Umständen schneller sind oder weniger Speicher brauchen. Wir werden daher sowohl die alten als auch die neuen Funktionen betrachten.

Der wichtigste Unterschied zwischen alten und neuen Dateifunktionen ist die Art, in der MS-DOS Dateien verwaltet. Mit alten Funktionen muß der Programmierer einen sogenannten »File-Control-Block« (kurz FCB) zur Verfügung stellen. Der Aufbau des FCB ist in Bild 1 grafisch dargestellt; wenn es um Verzeichnisse geht, ist auch diese Datenstruktur ein wichtiges Thema. Bei der Verwendung eines FCB gibt es Vor- und Nachteile. Auf einer Seite muß der Programmierer Speicher für einen FCB reservieren und Daten in dem von MS-DOS erwarteten Format in den FCB eintragen; eine Arbeit, die man gern umgeht, und die mit den neuen Funktionen entfällt. Andererseits kann man einige Informationen aus dem FCB auslesen, die sonst (unter Verwendung der neuen Funktionen) über andere Funktionen geholt werden müssen. Sie sehen vielleicht, warum die alten

MS-DOS-Funktionen immer noch eingesetzt werden.

Anstelle eines FCB brauchen die neuen Dateifunktionen lediglich die Adresse des Dateinamen oder aber einen sogenannten »Handle«. Der Name wird für das Öffnen beziehungsweise für das Erzeugen einer Datei gebraucht. Von diesen beiden Funktionen wird der »Handle« zurückgeliefert, der für alle weitere Aufrufe mit dieser Datei verwendet wird. Der Handle ist ein 16-Bit-Wert (also ein Wort), das der geöffneten Datei zugeordnet wird. Betrachten Sie den Handle einfach als logische Dateinummer, die die Datei identifiziert. Wenn die Datei geschlossen

wird, wird diese Zuordnung aufgelöst; der Handle kann anschließend einer anderen Datei zugeordnet werden.
Der größte Vorteil der neuen Funktionen ist, daß die Umlenkung von Ein- und Ausgabe über Handles realisiert wird, das heißt, nur Dateien, die mit neuen Funktionen geöffnet wurden, können umgelenkt werden. Unter diesen gibt es fünf Standard-Handles, die von MS-DOS zur Verfügung gestellt werden und die nicht speziell geöffnet werden müssen. Diese fünf Handles sind
0: Standardeingabe
1: Standardausgabe
2: Fehlerausgabe
3: Zusatzkanal
4: Drucker
Um es kurz zusammenzufassen: Mit den neuen Funktionen wird eine Datei mit der Adresse des Dateinamens geöffnet, beziehungsweise neu angelegt. Ein sogenannter »Handle« wird zurückgeliefert. Anschließend werden alle Funktionen mit diesem Handle aufgerufen, um Operationen auf der Datei auszuführen.
Um eine Datei mit einer alten Funktion zu öffnen oder zu erzeugen, muß man die entsprechende MS-DOS-Funktion mit der Adresse eines nicht geöffneten File Control Blocks (FCB) aufrufen. In einem nicht geöffneten FCB enthalten lediglich die ersten drei Felder (Laufwerk, Dateiname und Namenserverweiterung) Informatio-

Nummer	Name	Beschreibung
0FH	Datei eröffnen	Eröffnet eine bestehende Datei
10H	Datei erschließen	Erschließt eine offene Datei
13H	Datei löschen	Löscht eine bestehende Datei
14H	serielles Lesen	Liest von einer offenen Datei
15H	serielles Schreiben	Schreibt in einer offenen Datei
16H	Datei erzeugen	Erzeugt eine neue Datei
17H	Datei umbenennen	Nennt eine bestehende Datei um
21H	Direktes Lesen	Liest ein Feld aus einer Datei
22H	Direktes Schreiben	Schreibt ein Feld in eine Datei
23H	Dateigröße lesen	Liefert die Größe einer Datei
24H	Feld in Datei setzen	Verschiebt den Schreib/Lesezeiger in einer Datei
27H	Direktes Blocklesen	Liest Felder von einer Datei, beginnend beim Schreib/Lesezeiger
28H	Direktes Blockschreiben	Schreibt Felder in eine Datei, beginnend beim Schreib/Lesezeiger
3CH	Handle erzeugen	Erzeugt eine neue Datei oder öffnet eine bestehende Datei
3DH	Handle öffnen	Öffnet eine bestehende Datei
3EH	Handle schließen	Schließt eine offene Datei
3FH	Handle lesen	Liest aus einer Datei
40H	Handle schreiben	Schreibt in einer Datei
42H	Dateizeiger setzen	Verschiebt den Schreib/Lesezeiger in einer Datei
45H	Handle duplizieren	Erzeugt einen neuen Handle, der sich auf die gleiche Datei bezieht
46H	Handle kopieren	Kopiert einen bestehenden Handle in einen zweiten bestehenden Handle
5AH	Temporäre Datei erzeugen	Erzeugt eine temporäre Datei mit einem eindeutigen Namen
5BH	Neue Datei erzeugen	Erzeugt eine Datei, gibt einen Fehlercode zurück, falls die Datei bereits besteht

Tabelle 4. Dateifunktionen

nen; die restlichen Felder werden von der entsprechenden MS-DOS-Funktion versorgt.

Der Umgang mit Verzeichnissen ist unter MS-DOS ähnlich organisiert wie der Umgang mit Dateien

Wie bei den Dateifunktionen gibt es auch in der Gruppe der Verzeichnis-Funktionen alte und neue Funktionen. In Tabelle 5 sind alle Verzeichnis-Funktionen aufgelistet, wobei die alten Funktionen Nummern kleiner als 30hex haben. Da es nur zwei von diesen alten Funktionen gibt, zuerst zu diesen beiden.

Die beiden Funktionen 11hex und 12hex ermöglichen die Suche nach einem Dateinamen durch ein Verzeichnis. Wie alle alten Funktionen bekommen sie die Adresse eines FCB, allerdings in diesen besonderen Fällen muß es ein erweiterter FCB sein. In Bild 1 ist diese Erweiterung oben zu sehen. Wenn Sie die Felder in der Erweiterung anschauen, fällt Ihnen wahrscheinlich das Feld »Attribute« auf. In diesem Feld werden die Attribute der gesuchten Dateien nach dem Muster in Bild 2 eingetragen. Diese Attribute erlauben die Suche nach Verzeichnissen, versteckten Dateien und so weiter.

Außer den Attributen verlangt Funktion 11hex nur den üblichen nicht geöffneten FCB, der oben bereits beschrieben wurde. Nach einem Aufruf von der Funktion 11hex ist es möglich, weitere Dateien durch Funktion 12hex zu suchen. Dies ergibt jedoch nur dann Sinn, wenn der Dateiname im FCB ein oder mehrere Fragezeichen enthält, das heißt, wenn es mehrere passende Dateien geben könnte.

Wenn der Returncode im AL-Register 0 ist, wurde eine Datei gefunden. In diesem Fall hat die Funktion 11hex beziehungsweise 12hex die übrigen Felder im FCB gefüllt. Nun ist es möglich, die Informationen auszuwerten, wobei die meisten Felder einfach gelesen werden können. Das Datum und die Uhrzeit sind jedoch speziell verschlüsselt, können aber mit Hilfe von Bild 3 entschlüsselt werden. Listing 2 zeigt ein kleines Anwendungsbeispiel, aus dem die Funktionsweise deutlich wird.

Die neuen Funktionen sind im wesentlichen einfacher als die alten. Die meisten werden vom Namen her schon verständlich sein, lediglich die beiden Funktionen 4Ehex und 4Fhex, die analog zu den gerade diskutierten Funktionen 11hex und 1hex funktionieren, liefern eine Datenstruktur, die erklärungsbedürftig ist.

Nummer	Name	Beschreibung
11H	Ersten Eintrag suchen	Sucht den ersten passenden Eintrag in einem Verzeichnis
12H	Nächsten Eintrag suchen	Sucht den nächsten passenden Eintrag in einem Verzeichnis
39H	Verzeichnis erzeugen	Erzeugt ein neues Verzeichnis
3AH	Verzeichnis löschen	Löscht ein leeres Verzeichnis
3BH	Aktuelles Verzeichnis setzen	Setzt das aktuelle Verzeichnis
41H	Eintrag löschen	Löscht einen Eintrag in einem Verzeichnis (löscht eine Datei)
43H	Attribute lesen/schreiben	Liest oder schreibt die Attribute einer Datei
47H	Aktuelles Verzeichnis holen	Liest den Namen des aktuellen Verzeichnisses
4EH	Erste Datei finden	Findet die erste passende Datei
4FH	Nächste Datei finden	Findet die nächste passende Datei
56H	Eintrag ändern	Ändert den Dateinamen
57H	Datum/Uhrzeit lesen/schreiben	Liest oder schreibt das Datum und die Uhrzeit in einem Eintrag

Tabelle 5. Verzeichnisfunktionen

Nummer	Name	Beschreibung
00H	Programm beenden	Beendet das aufrufende Programm
26H	Neuen PSP erzeugen	Bereitet einen Prozeßstart vor
31H	Beenden, aber im Speicher bleiben	Beendet das aufrufende Programm, reserviert aber seinen Speicherbereich
31H	Prozeß behalten	Beendet einen Prozeß und reserviert seinen Speicherbereich
4B00H	Programm laden und starten	Lädt und startet ein Programm
4B03H	Overlay laden	Lädt ein Overlay (wird nicht gestartet)
4CH	Prozeß beenden	Beendet einen Prozeß
4DH	Returncode vom Unterprozeß holen	Liest Returncode, den ein Unterprozeß hinterlassen hat
62H	PSP holen	Holt die Adresse des PSP (Programm-Segment-Präfix)
Int20H	Programm beenden	Identisch zur Funktion 00H
Int27H	Beenden, aber im Speicher bleiben	Identisch zur Funktion 31H

Tabelle 6. Prozeßfunktionen

Bei allen Funktionsaufrufen enthält das Register AH die Funktionsnummer. Bei Rückkehr mit Fehler (Carry gesetzt) enthält AX bei den meisten neuen Routinen ab Version 2 einen Fehlercode. Nicht in der Tabelle aufgeführt sind die MS-DOS-Funktionen zur Netzwerkkontrolle (Microsoft Network).			
Funktion	00hex (0)	DOS 1	Terminate Program
Aktion	Beendet die Ausführung eines Programms		
Aufruf	AH: 0		
	CS: Segment-Adresse wie bei Programmstart		
Rückkehr	Funktion kehrt nicht zurück		
Funktion	01hex (1)	DOS 1	Read Keyboard and Echo
Aktion	Wartet auf Zeichen von Standardeingabe mit Echo und Prüfung auf Break		
Aufruf	AH: 1		
Rückkehr	AL: Zeichen		
Funktion	02hex (2)	DOS 1	Display Character
Aktion	Sendet Zeichen an die Standardausgabe		
Aufruf	AH: 2 DL: Zeichen		
Rückkehr	-		
Funktion	03hex (3)	DOS 1	Auxiliary Input
Aktion	Liest ein Zeichen von der seriellen Schnittstelle		
Aufruf	AH: 3		
Rückkehr	AL: Zeichen		
Funktion	04hex (4)	DOS 1	Auxiliary Output
Aktion	Gibt ein Zeichen an die serielle Schnittstelle aus		
Aufruf	AH: 4 DL: Zeichen		
Rückkehr	-		
Funktion	05hex (5)	DOS 1	Print Character
Aktion	Gibt ein Zeichen an den Drucker aus		
Aufruf	AH: 5 DL: Zeichen		
Rückkehr	-		

Tabelle 7. Alle MS-DOS-Funktionsaufrufe der Version 3.2 mit Registerbelegung

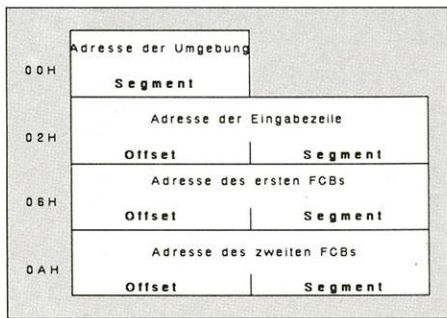


Bild 5. Der Parameterblock für die MS-DOS-Funktion 4Bhex

Diese Struktur sehen Sie in Bild 4. Sie merken, daß diese Struktur sehr ähnlich zum FCB ist. Auch die Attribute, das Datum und die Uhrzeit werden hier genauso wie im FCB kodiert, so können Sie die Beschreibungen direkt den Bildern 2 und 3 entnehmen. Der Dateiname wird hier allerdings anders geschrieben. Hier wird der Name genauso zurückgegeben, wie Sie ihn in einem Kommando angeben würden: ohne Leerzeichen und mit einem Punkt zwischen Hauptnamen und Namensweiterung.

MS-DOS-Prozesse: Programme von einem Programm aus starten

Die letzte hier zu besprechende Gruppe von MS-DOS-Funktionen bilden die Prozeßfunktionen. Diese sind in Tabelle 6 aufgeführt, wobei die alten Funktionen wie üblich Nummern kleiner als 30hex haben. Zusätzlich gibt es hier zwei Interrupts, die auch als alte Funktionen anzusehen sind. In dieser Gruppe ist es der Fall, daß die alten Funktionen wirklich nicht mehr verwendet werden sollen. Der einzige Grund dafür wäre, Kompatibilität zu MS-DOS-Versionen älter als 2.0 zu gewährleisten, jedoch werden diese 1.x Versionen überhaupt nicht mehr benutzt.

Wenn Sie sich nun fragen, wozu diese Prozeß-Funktionen gut sind, können wir ein wichtiges Beispiel nennen: Diese Funktionen werden von »command.com« benutzt, um alle anderen Programme zu starten und zu verwalten.

Unter MS-DOS hat der Begriff »Prozeß« fast die gleiche Bedeutung wie »Programm«. Der Unterschied ist rein semantisch: Ein Programm ist ablauffähig, das Wort sagt aber nicht aus, ob das Programm zu diesem Zeitpunkt läuft. Im Gegensatz dazu wird der Begriff »Prozeß« verwendet, um anzudeuten, daß der Prozeß zu diesem Zeitpunkt abläuft. Ein gerade laufendes Programm ist also in diesem Zusammenhang ein Prozeß. Daher gibt es selbstverständlich eine MS-DOS-Funktion (Nummer 4Bhex),

Funktion	06hex (6)	DOS 1	Direct Console I/O
Aktion	Direkte Ein-/Ausgabe. Wenn DL = 255 bei Aufruf, dann wird ein Zeichen von Standardeingabe geholt, sonst wird Zeichen in DL an Standardausgabe gesendet		
Aufruf	AH: 6	DL: 255 oder Zeichen	
Rückkehr	AL = Zeichen, falls DL = 255 bei Aufruf. Falls kein Zeichen vorhanden, dann AL = 0 und Zero-Flag gesetzt		
Funktion	07hex (7)	DOS 1	Direct Console Input
Aktion	Wartet auf Zeichen von Standardeingabe, ohne Echo und Break-Prüfung		
Aufruf	AH: 7		
Rückkehr	AL: Zeichen		
Funktion	08hex (8)	DOS 1	Read Keyboard
Aktion	Wartet auf Zeichen von Standardeingabe, mit Echo, aber ohne Break-Prüfung		
Aufruf	AH: 8		
Rückkehr	AL: Zeichen		
Funktion	09hex (9)	DOS 1	Display String
Aktion	Sendet einen String an Standardausgabe. Der String muß durch ein Dollarzeichen abgeschlossen sein		
Aufruf	AH: 9	DS:DX: Zeiger auf den Anfang des Strings	
Rückkehr	-		
Funktion	0Ahex (10)	DOS 1	Buffered Keyboard Input
Aktion	Liest einen String von Standardeingabe in einen Puffer. Das erste Byte in diesem Puffer muß vor Aufruf die maximal mögliche Stringlänge enthalten, das zweite Byte enthält nach dem Aufruf die tatsächliche Länge des Strings		
Aufruf	AH: 10	DS:DX: Zeiger auf Eingabepuffer	
Rückkehr	-		
Funktion	0Bhex (11)	DOS 1	Check Keyboard Status
Aktion	Stellt fest, ob ein Zeichen von Standardeingabe verfügbar ist		
Aufruf	AH: 11		
Rückkehr	AL = 0, falls kein Zeichen verfügbar, sonst 255		
Funktion	0Chex (12)	DOS 1	Flush Buffer and Read Keyboard
Aktion	Löscht Eingabepuffer und führt dann die in AL spezifizierte Eingabefunktion aus		
Aufruf	AH: 12	AL: 1,6,7,8 oder 10, falls Aufruf gewünscht	
Rückkehr	AL = 0, falls nur Eingabepuffer gelöscht, aber keine andere Funktion aufgerufen wird		
Funktion	0Dhex (13)	DOS 1	Reset Disk
Aktion	Zurücksetzen des Disketten-Systems		
Aufruf	AH: 13		
Rückkehr	-		
Funktion	0Ehex (14)	DOS 1	Select Disk
Aktion	Aktuelles Laufwerk bestimmen		
Aufruf	AH: 14	DL: Laufwerksnummer (0 = A, 1 = B, etc.)	
Rückkehr	AL: Anzahl der vorhandenen logischen Laufwerke		
Funktion	0Fhex (15)	DOS 1	Open File
Aktion	Datei öffnen (CP/M-kompatible Funktion)		
Aufruf	AH: 15	DS:DX: Zeiger auf File Control Block	
Rückkehr	AL = 0, falls Directory-Eintrag für Datei gefunden wurde, sonst AL = 255		
Funktion	10hex (16)	DOS 1	Close File
Aktion	Datei öffnen (CP/M-kompatible Funktion)		
Aufruf	AH: 16	DS:DX: Zeiger auf File Control Block	
Rückkehr	AL = 0, falls Directory-Eintrag für Datei gefunden wurde, sonst AL = 255		
Funktion	11hex (17)	DOS 1	Search for First Entry
Aktion	Ersten Dateinamen suchen (CP/M-kompatible Funktion)		
Aufruf	AH: 17	DS:DX: Zeiger auf File Control Block	
Rückkehr	AL = 0, falls Directory-Eintrag für Datei gefunden wurde, sonst AL = 255		
Funktion	12hex (18)	DOS 1	Search for Next Entry
Aktion	Nächsten Dateinamen suchen (CP/M-kompatible Funktion)		
Aufruf	AH: 18	DS:DX: Zeiger auf File Control Block	
Rückkehr	AL = 0, falls Directory-Eintrag für Datei gefunden wurde, sonst AL = 255		
Funktion	13hex (19)	DOS 1	Delete File
Aktion	Datei löschen (CP/M-kompatible Funktion)		
Aufruf	AH: 19	DS:DX: Zeiger auf File Control Block	
Rückkehr	AL = 0, falls Directory-Eintrag für Datei gefunden wurde, sonst AL = 255		
Funktion	14hex (20)	DOS 1	Sequential Read
Aktion	Liest Record aus einer Datei (CP/M-kompatible Funktion)		
Aufruf	AH: 20	DS:DX: Zeiger auf File Control Block	
Rückkehr	AL = 0: Daten erfolgreich gelesen AL = 1: Dateiende erreicht AL = 2: DTA zu klein AL = 3: letzter Record in Datei		
Funktion	15hex (21)	DOS 1	Sequential Write
Aktion	Schreibt Record in Datei (CP/M-kompatible Funktion)		
Aufruf	AH: 21	DS:DX: Zeiger auf File Control Block	
Rückkehr	AL = 0: Daten erfolgreich geschrieben AL = 1: Diskette voll AL = 2: DTA zu klein		

Tabelle 7. Alle MS-DOS-Funktionsaufrufe der Version 3.2 mit Registerbelegung

ProSoft-Preise liegen richtig!

☎ 0261/40 47-1 • **Tx** 862476 PSOFT • Telefax 0261/40 47-252

Wir suchen ständig günstige Einkaufsquellen für die angebotenen und neue innovative Produkte.

Günstige Möglichkeit der Finanzierung durch Ratenkredit. Fordern Sie die Unterlagen an.

Olivetti Olivetti Olivetti

Die neue Serie 200 jetzt auch mit 3,5" Diskettenlaufwerken und voll IBM-kompatibel.

M 240/55 G

8086 mit 10 MHz, 640 KB Hauptspeicher 2 Diskettenlaufwerke a' 360 KB, serielle und parallele Schnittstelle, Echtzeituhr, OGC-Controller, Tastatur II, Monitor, MS-DOS 3.2 und GW-Basic **3298.-**

M 240/0520 G2 (2 Diskettenlaufwerke) wie M240/55G, jedoch mit 20 MB Festplatte **3898.-**

Natürlich liefern wir das gesamte Angebot des M240. Bitte erfragen Sie unsere EGA- und Positiv-Versionen!

NEU! M280

80286 mit 12 MHz Taktfrequenz, 1 MB Hauptspeicher, 1 Diskettenlaufwerk 1,2 MB, 1 Festplatte 20 MB, Monitor, Tastatur, MS-DOS/GW-Basic **5998.-**

M280-40

wie M280, jedoch mit 40 MB Festplatte **6298.-**

M380C/0540E komplett mit Monitor, Tastatur, MS-DOS/GW-Basic **10398.-**

Commodore Commodore

PC-10 III

8088-2 mit 4,77/7,16 und 9,54 MHz Taktfrequenz, 640 KB Hauptspeicher, parallele und serielle Schnittstelle, Maus-Interface, AGA - Grafikadapter (Monochrom und Farbe), Echtzeituhr, 2 Diskettenlaufwerke a'360 KB, MF-Tastatur, Monitor, MS-DOS 3.2 und GW-Basic **1798.-**

Beim PC-10 III können Lieferzeiten entstehen, jetzt anrufen!

PC-10 III 2/20 wie PC-10 III, jedoch mit 20 MB Festplatte **2398.-**

PC-10 III 2/30 wie PC-10 III, jedoch mit 30 MB Festplatte **2498.-**

PC-10 III 2/50 wie PC-10 III, jedoch mit 50 MB Festplatte **2798.-**

PC-20 III

wie PC-10 III, jedoch nur 1 Diskettenlaufwerk 360 KB und 20 MB Festplatte **2598.-**

Neu! Commodore 386 PC- 60/40

Commodore PC 60/40 80386 CPU mit 4,77/6/8/10/12 und 16 MHz Takt umschaltbar, 1 MB Hauptspeicher, 2 serielle und parallele Schnittstellen, EGA-Grafikadapter, 1 Disk. 1,2 MB, 1 Festplatte 40 MB, MF-Tastatur Monitor 14", MS-DOS 3.2 und GW - Basic **9696.-**

Commodore PC 60/122

wie PC 60/40 jedoch mit 122 MB Festpl. **11498.-**

Amiga 2000 mit Monitor 1084 **2768.-**
Amiga 500 **1148.-**

Tandon - Tandon - Tandon

PCA 20 plus **4798.-**

PCA 40 plus	6398.-	PCA 70 plus	7598.-
Target 20	4398.-	Target 40	5998.-
Target 20 plus	5198.-	Target 40 plus	6798.-
PAC 286	4398.-	PAC 286 plus	4798.-
Data-Pac	798.-	PAC Floppy	998.-

Amstrad Amstrad Amstrad

Bitte erfragen Sie unsere verschiedenen Amstrad-Konfigurationen (PC-1512, PC-1640). Bitte anrufen!

Schneider PC - Schneider PC

PC 2640 MM **3998.-** PC 2640 EM **4998.-**

Plantron Plantron Plantron

NEU! Alle Plantron-Modelle werden jetzt mit dem neuen MS-DOS 3.3 geliefert.

PT LC **1245.-**
PT LC/20 mit 20 MB Festplatte **1898.-**
PT LC/30 mit 30 MB Festplatte **1948.-**

PT-XT **1688.-**
PT-XT 2/20 mit 20 MB Festplatte **2348.-**
PT-XT 2/30 mit 30 MB Festplatte **2398.-**

PT-ST 80286 mit 8/10 MHz, 640 KB Hauptspeicher, Herkules kompatible Grafikkarte mit paralleler Schnittstelle, 1 Diskettenlaufwerk 1,2 MB, MF -Tastatur, MS-DOS und PBasic sowie umfangreiche Bedienungsanleitung **2198.-**

PT-AT wie PT-ST, jedoch mit zusätzlicher Multi I/O-Karte (die mit einer seriellen und parallelen Schnittstelle sowie Game - Port bestückt ist), AT-FD/HD-Controller **2598.-**

PT-AT/20 wie PT-AT, mit 20 MB Festplatte **3148.-**

PT-AT/30 wie PT-AT, mit 30 MB Festplatte **3298.-**

PT-AT/40 wie PT-AT, mit 40 MB Festplatte **3498.-**

PT-AT/80 wie PT-AT, mit 80 MB Festplatte **4098.-**

PT-AT/122 wie PT-AT, mit 122 MB Festplatte **4398.-**

PT-286 AT (Baby-AT) mit 64 MB Festplatte und Super EGA-Karte **+**
Hitachi Multi 560 EGA Autoscan mit Analog + TTL-Anschluß kompl. nur **4748.-**

PT-HT/2 80386 mit 16 MHz Taktfrequenz, 1 MB Hauptspeicher, 1 Diskettenlaufwerk 1,2 MB, 1 Festplatte 32 MB, serielle und parallele Schnittstelle, Herkules kompatible Grafikkarte, MF - Tastatur, MS - DOS und PBasic **5698.-**

PT-HT wie PT-HT/2, jedoch mit Super EGA-Karte und 64 MB Festplatte **6298.-**

PT-HT/80 wie PT-HT/2, jedoch mit Super EGA-Karte und 80 MB Festplatte **6698.-**

PT-HT/122 wie PT-HT/2, jedoch mit Super EGA-Karte und 122 MB Festplatte **7198.-**

Seagate Festplatten

20 MB Festplattenkit **528.-**

30 MB Festplattenkit **578.-**

30 MB Festplattenkit **698.-**

ST-138 incl. Contr. + Kabels., 3,5", 40ms

20 MB (ST 225) **418.-** 30 MB (ST 238) **458.-**

65 MB (ST 277R) **858.-** 122 MB (ST-4144R) **1798.-**

40 MB (ST 251/1) **898.-** 80 MB (ST 4096) **1398.-**

ST 251-0 40 MB, 40ms **698.-**

20 MB (ST 125-0) **498.-** 20 MB (ST 125-1) **598.-**

50 MB (ST 157R-0) **798.-** 50 MB (ST 157R-1) **888.-**

Take Ten Diskettenlaufwerk 10 MB intern **1298.-**

Take Ten Diskettenlaufwerk 10 MB extern **1598.-**

Tape Streamer 40 MB "Alloy APT-40" jetzt Quick-Tape kompatibel incl. DC-2000 Cassette nur **748.-**

52 MB "Wangtek FAD 5000" für XT oder AT **858.-**

Datencassette DC-2000 (für APT-40) **68.-**

Datencassette CC-600A (für FAD 5000) **78.-**

Co-Prozessoren

8087 (5 MHz) **198.-** 8087 (8 MHz) **329.-**

8087 (10 MHz) **398.-** 80287 (6 MHz) **329.-**

80287 (8 MHz) **489.-** 80287 (10 MHz) **598.-**

80387-16 **1098.-** 80387-20 **1998.-**

Monitore

NEC Multisync II **1398.-** NEC Multisync I **1298.-**

NEC Multisync GS **498.-** NEC Multisync Plus **2098.-**

Mitsubishi Autoscan EUM-1471 A **1298.-**

14" ADI kompatibler Monitor, grün oder bernstein **248.-**

14" Flat-Screen Monitor, bernstein oder s/w **248.-**

Hitachi Multi 560 mit Analog und TTL-Anschluss **1298.-**

EGA/VGA Grafik - Adapter

EGA Wonder Enhanced EGA mit VGA **398.-**

VIP - VGA Karte von ATI **598.-**

VEGA de Luxe Autoswitch EGA Karte **598.-**

Video Seven VEGA VGA **648.-**

Paradise EGA (80 Zeichen) ohne Autoswitch nur **238.-**

Paradise EGA Autoswitch (80 Zeichen) **268.-**

Paradise EGA Autoswitch (132 Zeichen) **398.-**

Paradise Plus Card für 8-Bit-Steckplatz **798.-**

Paradise Professional Card für 16-Bit-Steckplatz **998.-**

NEU! Original Genoa Super EGA Hires + (800 x 600, mit VGA) **398.-**

Paradise EGA Autoswitch (80 Zeichen) **+**

Hitachi Multi 560 EGA Autoscan kompl. nur **1498.-**

Atari - Atari - Atari

Atari PC 1 incl. Maus + Software, MS - DOS 3.21

Basic: GEM, Startup, Desktop, GEMwrite, GEMpaint, incl. Monitor **1398.-**

Atari 1040 STF, Tastatur, 1024KB RAM, 192KB ROM, integrierte Floppy 720 KB, Monochrom-Monitor SM 124, Maus, Basic **1498.-**

Brother - Brother - Brother

M-1409 849.- M-1509 998.- M-4018 S 2998.-

M-1709 1198.- M-1724 L 1448.- M-4018 C 3198.-

Star - Star - Star

LC-10 548.-

incl. Centronics- oder Commodore Schnittstelle

Epson - Epson - Epson

LQ-850 1248.- LQ-1050 1648.-

LQ-500 798.- FX-800 918.- FX-1000 1188.-

LQ-2500 2438.- SQ-2500 3188.- LX-800 529.-

EX-800 1298.- EX-1000 1628.- LX-800 VC/P 619.-

GQ-3500 3978.-

Einzelblatteinzug für LQ-850 **328.-**

Einzelblatteinzug für LQ-1050 **398.-**

NEC - NEC - NEC - NEC - NEC

P 2200 sofort lieferbar! nur **798.-**

P 7 1298.- P 6 1098.-

P 7 color 1548.- P 6 color 1298.-

P 7 seriell 1648.- P 6 seriell 1398.-

P 7 seriell color 2298.- P 6 seriell color 1848.-

Pin-Feed-Traktor P7 228.- Pin-Feed-Traktor P6 109.-

Bidirekt. Traktor P7 348.- Bidirekt. Traktor P6 298.-

Cut-Sheet-Feeder P7 698.- Cut-Sheet-Feeder P6 598.-

Kyocera - Laserdrucker

F-1000 4999.- F-2200 10698.-

F-1200 7498.- F-3000 14898.-

Citizen - Citizen - Citizen

LSP-120D comm. oder paral. Schnittstelle **394.-**

Seikosha - Seikosha - Seikosha

SL-80 AI nur **768.-** SL-80 VC **768.-**

Okidata - Okidata - Okidata

ML-182 parallel **448.-** ML-192 Elite I/E **798.-**

ML-292 Elite **948.-** Personality Mod.f.292 **268.-**

ML-294 Elite **1498.-** Personality Mod.f.294 **268.-**

ML-393 **2298.-** Emulationsm.f.393 **148.-**

ML-393 color **2498.-** Emulationsm.f.393 **148.-**

Okimate OM-20 **398.-**

Panasonic - Panasonic

KX-F 1081 **478.-**

Weitere Panasonic Produkte lieferbar. Es können kurzzeitige Lieferengpässe entstehen. Bitte rufen Sie an!

ProSoft GmbH

Bogenstraße 51-53, Postfach 207, D-5400 Koblenz-Goldgrube, Telefon (0261) 4047-1, Telex 862476, Telefax (0261) 4047-252

Alle Preise zzgl. 10,- DM Versandkosten pro Paket. Lieferung per Nachnahme oder Vorkassascheck - Versandkosten Ausland DM 40,- pro Paket. Lassen Sie sich keinen Bären auflockern! ProSoft liefert Original-Produkte der führenden Hersteller. Überzeugen Sie sich selbst durch Abholung der Ware in unseren Verkaufsräumen in Koblenz. Wir gewähren Ihnen bei Barzahlung (kein Scheck!) 2% Skonto auf alle Preise, was vielleicht schon zur Deckung Ihrer Reisekosten ausreicht. Einige unserer Vorlieferanten liefern Produkte ohne die Seriennummer des Herstellers. In diesem Fall übernehmen wir anstelle der Herstellergarantie die unbeschränkte gesetzliche Gewährleistung.

Filiale München Theresienstraße 56, 8000 München 2, Tel. 0 89/2 80 93 89 direkt bei der technischen Hochschule. Bitte beachten Sie, daß nicht ständig sämtliche Ware in unserer Filiale München vorrätig ist. Rufen Sie an!

Funktion	16hex (22)	DOS 1	Create File
Aktion	Erzeugt neue Datei (CP/M-kompatible Funktion)		
Aufruf	AH: 22	DS:DX: Zeiger auf File Control Block	
Rückkehr	AL = 0, falls leerer Directory-Eintrag für Datei gefunden wurde, sonst AL = 255		
Funktion	17hex (23)	DOS 1	Rename File
Aktion	Datei umbenennen (CP/M-kompatible Funktion)		
Aufruf	AH: 23	DS:DX: Zeiger auf modifizierten FCB	
Rückkehr	AL = 0, falls Directory-Eintrag für Datei gefunden wurde, sonst AL = 255		
Funktion	18hex (24)	Reserviert	
Funktion	19hex (25)	DOS 1	Get Current Disk
Aktion	Liefert Nummer des aktuellen Laufwerks		
Aufruf	AH: 25		
Rückkehr	AL: aktuelles Laufwerk (0 = A, 1 = B, etc.)		
Funktion	20hex (26)	DOS 1	Set Disk Transfer Address
Aktion	DTA-Bereich festlegen		
Aufruf	AH: 26	DS:DX: Disk Transfer Adresse (DTA)	
Rückkehr	–		
Funktion	1Bhex (27)	DOS 1	Get Default Drive Data
Aktion	Informationen über das voreingestellte Laufwerk aus der File Allocation Table (FAT) holen		
Aufruf	AH: 27		
Rückkehr	AL: Anzahl Sektoren pro Block CX: Anzahl Bytes pro Sektor DX: Anzahl Blöcke pro Diskette DS:BX: Zeiger auf FAT-ID-Byte		
Funktion	1Chex (28)	DOS 1	Get Drive Data
Aktion	Informationen über ein beliebiges Laufwerk aus der File Allocation Table (FAT) holen		
Aufruf	AH: 28	DL: Laufwerk (0 = aktuelles, 1 = A, 2 = B, etc.)	
Rückkehr	AL: 255, wenn Laufwerksnummer ungültig, sonst: Anzahl Sektoren pro Block CX: Anzahl Bytes pro Sektor DX: Anzahl Blöcke pro Diskette DS:BX: Zeiger auf FAT-ID-Byte		
Funktion	1Dhex (29)	Reserviert	
Funktion	1Ehex (30)	Reserviert	
Funktion	1Fhex (31)	Reserviert	
Funktion	20hex (32)	Reserviert	
Funktion	21hex (33)	DOS 1	Random Read
Aktion	Record wahlfrei lesen (CP/M-kompatible Funktion)		
Aufruf	AH: 33	DS:DX: Zeiger auf File Control Block	
Rückkehr	AL = 0: Lesezugriff erfolgreich durchgeführt AL = 1: Dateiende erreicht, Record leer AL = 2: DTA zu klein AL = 3: Dateiende erreicht, Record nicht vollständig		
Funktion	22hex (34)	DOS 1	Random Write
Aktion	Record wahlfrei schreiben (CP/M-kompatible Funktion)		
Aufruf	AH: 34	DS:DX: Zeiger auf File Control Block	
Rückkehr	AL = 0: Schreibzugriff erfolgreich durchgeführt AL = 1: Diskette voll AL = 2: DTA zu klein		
Funktion	23hex (35)	DOS 1	Get File Size
Aktion	Berechnet Dateilänge und trägt sie im FCB ein		
Aufruf	AH: 35	DS:DX: Zeiger auf File Control Block	
Rückkehr	AL = 0, falls Directory-Eintrag für Datei gefunden wurde, sonst AL = 255		
Funktion	24hex (36)	DOS 1	Set Relative Record
Aktion	Setzt FCB-Feld für relativen, wahlfreien Zugriff		
Aufruf	AH: 36	DS:DX: Zeiger auf File Control Block	
Rückkehr	–		
Funktion	25hex (37)	DOS 1	Set Interrupt Vector
Aktion	Interrupt-Vektor auf neuen Wert setzen		
Aufruf	AH: 37	AL: Interrupt-Nummer	
	DS:DX: Adresse der Interrupt-Routine		
Rückkehr	–		
Funktion	26hex (38)	DOS 1	Create New PSP
Aktion	Neuen Program Segment Prefix (PSP) setzen		
Aufruf	AH: 38	DX: Segment-Adresse des neuen PSP	
Rückkehr	–		
Funktion	27hex (39)	DOS 1	Random Block Read
Aktion	Einen oder mehrere Records wahlfrei lesen		
Aufruf	AH: 39	DS:DX: Zeiger auf File Control Block	
	CX: Anzahl zu lesender Records		
Rückkehr	AL = 0: Lesezugriff erfolgreich durchgeführt AL = 1: Dateiende erreicht, Record leer AL = 2: DTA zu klein AL = 3: Dateiende erreicht, Record nicht vollständig CX: Anzahl gelesener Records		

Tabelle 7. Alle MS-DOS-Funktionsaufrufe der Version 3.2 mit Registerbelegung

die ein Programm lädt und startet (wodurch es zum Prozeß wird). Diese Funktion braucht die Adresse des Namens der Datei, worin das Programm enthalten ist, und die Adresse eines Parameterblocks. Dieser Block ist in Bild 5 zu sehen. Um einen Prozeß starten zu können, muß MS-DOS genug Speicher für das Programm zur Verfügung haben; sonst liefert MS-DOS die Meldung »zu wenig Speicher« zurück und verweigert es, das Programm zu starten.

Ein Prozeß kann sich in einer von zwei Arten beenden. Mit der Funktion 4Chex beendet sich ein Prozeß und gibt die Kontrolle zurück an den Vater-Prozeß, das heißt an den, der ihn (mittels Funktion 4Bhex) ins Leben gerufen hat. Wenn ein Prozeß die Funktion 4Chex aufruft, wird er vollständig aus dem Speicher entfernt. Wahlweise kann der zu löschende Prozeß einen Returncode an den Vater-Prozeß übergeben, um seinen Erfolg oder Mißerfolg bekanntzugeben. Der übergeordnete Prozeß kann anschließend diesen Returncode mit der Funktion 4Dhex holen.

Die andere Art, einen Prozeß zu beenden (über Funktion 31H beziehungsweise über den alten Interrupt 27hex) erlaubt es, daß ein Prozeß sich beendet, aber trotzdem im Speicher bleibt (»Terminate and Stay Resident«). Diese zunächst sinnlos erscheinende Funktion hat in Wahrheit eine große Bedeutung – sie ist die Grundlage für alle speicherresidenten Programme. Mehr dazu am Schluß dieses Kurses.

Um den Umgang mit Prozessen etwas zu beleuchten, finden Sie in Listing 3 als Beispiel ein Turbo-Pascal-Programm, das nur zählt. Wenn aber ein Aufrufzeichen eingegeben wird, startet das Programm einen neuen Kommandoprozessor »command.com« und ermöglicht es damit, das der Benutzer beliebige MS-DOS-Kommandos ausführt. Mit dem Befehl »Exit« kehrt der Anwender dann zum Pascal-Programm zurück.

Sie kennen bestimmt Programme, die die Eingabe von DOS-Kommandos erlauben – jetzt wissen Sie auch, wie dies realisiert wird. Sie können das Beispielprogramm nehmen und beliebige Funktionen einfügen. Solange Sie gelegentlich die Tastatur überprüfen und auf das Aufrufzeichen wie im Musterprogramm reagieren, werden alle MS-DOS-Kommandos auch in Ihrem Programm verfügbar sein.

Damit ist unser Überblick der MS-DOS-Funktionen zu Ende. Zu guter Letzt eine kleine Erinnerung: Ähnlich wie bereits bei der BIOS-Bibliothek aus dem zweiten Teil gilt auch hier, daß bei der Verwendung von Funktionsaufrufen äußerste Vorsicht gebo-

ten ist. Experimentieren Sie bitte mit solchen Programmen niemals, ohne den Quelltext vorher zu sichern, und hüten Sie sich wohlweislich, Disket-

tenfunktionen an Originalen oder an Ihrer Festplatte auszuprobieren. Auf den nächsten Seiten finden Sie eine komplette Übersicht über alle MS-

DOS-Funktionsaufrufe als Referenz.

(Thomas Little/ev)

Funktion 28hex (40) DOS 1 Random Block Write Aktion Einen oder mehrere Records wahlfrei schreiben Aufruf AH: 40 DS:DX: Zeiger auf File Control Block Rückkehr CX: Anzahl zu schreibender Records AL = 0: Schreibzugriff erfolgreich durchgeführt AL = 1: Diskette voll AL = 2: Segmentende erreicht CX: Anzahl geschriebener Records	Funktion 2Ehex (46) DOS 1 Set/Reset Verify Flag Aktion Automatisches Verify nach Schreibzugriffen ein/aus Aufruf AH: 46 AL: 0 für Verify aus, 1 für Verify ein Rückkehr -
Funktion 29hex (41) DOS 1 Parse Filename Aktion Dateinamen der Form "A:NAME.EXT" prüfen und FCB-Eintrag daraus herstellen. Wildcards werden unterstützt Aufruf AH: 41 AL: Bitmuster Bit 0 = 0: Stop, wenn Delimiterzeichen = 1: Führende Delimiter ignorieren Bit 1 = 0: Laufwerksnummer auf Null setzen, falls kein Laufwerksbezeichner im Namen vorhanden = 1: Laufwerksnummer nicht ändern Bit 2 = 0: Dateiname auf acht Leerzeichen setzen, falls kein Dateiname angegeben = 1: Dateiname nicht ändern Bit 3 = 0: Namenserweiterung auf drei Leerzeichen setzen, falls keine Namenserweiterung angegeben = 1: Namenserweiterung nicht ändern Rückkehr DS:SI: Zeiger auf zu untersuchenden Namen ES:DI: Zeiger auf FCB-Puffer AL = 0: Keine Wildcards im Namen AL = 1: Wildcards gefunden AL = 255: ungültiges Laufwerk DS:SI: Zeiger auf erstes Byte nach geprüften Namen ES:DI: Zeiger auf File Control Block	Funktion 2Fhex (47) DOS 2 Get Disk Transfer Adress Aktion Start des gegenwärtigen DTA-Bereichs holen Aufruf AH: 47 Rückkehr ES:BX: Zeiger auf DTA-Bereich
Funktion 2Ahex (42) DOS 1 Get Date Aktion Datum holen Aufruf AH: 42 Rückkehr CX: Jahr (1980 bis 2099) DH: Monat (1 bis 12) DL: Tag (1 bis 31) AL: Wochentag (0 = Sonntag, 1 = Montag, etc.)	Funktion 30hex (48) DOS 2 Get MS-DOS Version Number Aktion MS-DOS-Versionsnummer und Seriennummer holen Aufruf AH: 48 Rückkehr AL: Hauptversionsnummer AH: Nebenversionsnummer BH: OEM-Seriennummer BL: CX: 24-Bit-Benutzer-Seriennummer
Funktion 2Bhex (43) DOS 1 Set Date Aktion Datum setzen Aufruf AH: 43 CX: Jahr (1980 bis 2099) DH: Monat (1 bis 12) DL: Tag (1 bis 31) Rückkehr AL = 0, falls gültiges Datum, sonst AL = 255	Funktion 31hex (49) DOS 2 Keep Process Aktion Programm beenden, aber resident im Speicher behalten Aufruf AH: 49 AL: Return-Code DX: benötigter Speicherbereich in Paragraphen Rückkehr Funktion kehrt nicht zurück
Funktion 2Chex (44) DOS 1 Get Time Aktion Uhrzeit holen Aufruf AH: 44 Rückkehr CH: Stunde (0 bis 23) CL: Minute (0 bis 59) DH: Sekunde (0 bis 59) DL: Millisekunde (0 bis 99)	Funktion 32hex (50) Reserviert
Funktion 2Dhex (45) DOS 1 Set Time Aktion Uhrzeit setzen Aufruf AH: 45 CH: Stunde (0 bis 23) CL: Minute (0 bis 59) DH: Sekunde (0 bis 59) DL: Millisekunde (0 bis 99) Rückkehr AL = 0, falls gültige Uhrzeit, sonst AL = 255	Funktion 33hex (51) DOS 2 Control-C Check Aktion Prüft oder setzt den Status der Break-Prüfung Aufruf AH: 51 AL: 0 für Status holen, 1 für Status setzen DL: (falls AL = 1) 0 für Break-Prüfung aus, 1 für ein Rückkehr DL: (falls AL = 0 bei Aufruf) 0 = Break-Prüfung aus, 1 = Break-Prüfung ein AL = 255, falls Fehler (AL weder 0 noch 1 bei Aufruf)
	Funktion 34hex (52) Reserviert
	Funktion 35hex (53) DOS 2 Get Interrupt Vector Aktion Adresse einer Interrupt-Routine holen Aufruf AH: 53 AL: Nummer des Interrupts Rückkehr ES:BX: Zeiger auf Interrupt-Routine
	Funktion 36hex (54) DOS 2 Get Disk Free Space Aktion Disketteninformationen holen Aufruf AH: 54 DL: Laufwerk (0 = aktuelles, 1 = A, 2 = B, etc.) Rückkehr AX: -1, falls ungültige Laufwerksangabe, sonst Sektoren pro Block BX: Anzahl freier Blöcke CX: Anzahl Bytes pro Sektor DX: Anzahl Blöcke pro Laufwerk
	Funktion 37hex (55) Reserviert
	Funktion 38hex (56) DOS 2 Get Country Data Aktion Landesspezifische Informationen holen Aufruf AH: 56 DS:DX: Zeiger auf 32 Byte freien Speicher AL: 0 = aktuelles Land 1 bis 254 = Landescodes 255: BX enthält Landescodes BX: (wenn AL = 255) Landescodes 255 oder höher Rückkehr Carry gesetzt: Fehler mit Nummer AX, sonst: BX: Landescodes DS:DX: Zeiger auf 32-Byte-Datenbereich mit landesspezifischen Informationen
	Funktion 38hex (56) DOS 2 Set Country Data Aktion Landescodes setzen Aufruf AH: 56 DS:DX: -1 (0FFFFH) AL: 0 = aktuelles Land 1 bis 254 = Landescodes 255: BX enthält Landescodes BX: (wenn AL = 255) Landescodes 255 oder höher Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst OK
	Funktion 39hex (57) DOS 2 Create Directory Aktion Anlegen eines Subdirectories Aufruf AH: 57 DS:DX: Zeiger auf Pfadnamen Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst OK
	Funktion 3Ahex (58) DOS 2 Remove Directory Aktion Subdirectory löschen Aufruf AH: 58 DS:DX: Zeiger auf Pfadnamen Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst OK

Tabelle 7. Alle MS-DOS-Funktionsaufrufe der Version 3.2 mit Registerbelegung

Funktion 3Bhex (59) DOS 2 Change Current Directory Aktion Subdirectory wechseln Aufruf AH: 59 DS:DX: Zeiger auf Pfadnamen Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst OK	Funktion 44hex (68) DOS 2 IOCTL Data Aktion Gerätedaten holen oder setzen Aufruf AH: 68 AL: 0 = Gerätedaten holen, 1 = Daten setzen BX: Dateinummer des Gerätes DX: (wenn AL = 1) Gerätedaten Bit 0: Standard-Eingabegerät Konsole Bit 1: Standard-Ausgabegerät Konsole Bit 2: NUL-Gerät (Datensenke) Bit 3: Gerät hat Uhrenfunktion Bit 4: Reserviert Bit 5: Gerät prüft nicht auf Kontrollzeichen Bit 6: Kein Dateiende bei Eingabe Bit 7 bis 13: Reserviert Bit 14: Gerät kann Kontrollzeichenketten bei Lesen und Schreiben verarbeiten Bit 15: Reserviert Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: DX: (wenn AL = 0 bei Aufruf) Gerätedaten wie oben
Funktion 3Chex (60) DOS 2 Create Handle Aktion Datei anlegen Aufruf AH: 60 DS:DX: Zeiger auf Pfadnamen CX: Dateiattribut Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: AX: Dateinummer	Funktion 44hex (68) DOS 2 IOCTL Character Aktion Steuerdaten an Gerät senden oder vom Gerät empfangen Aufruf AH: 68 AL: 2 = Daten senden, 3 = Daten empfangen BX: Dateinummer des Gerätes CX: Anzahl der zu sendenden/empfangenden Bytes DS:DX: Zeiger auf Datenpufferbereich Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: AX: Anzahl der übertragenen Bytes
Funktion 3Dhex (61) DOS 2 Open Handle Aktion bestehende Datei öffnen Aufruf AH: 61 DS:DX: Zeiger auf Pfadnamen AL: Zugriffscod Bits 0 bis 3: Access-Code Bits 4 bis 6: Sharing Mode Bit 7: Inherit Bit Access-Code: 0000 = nur Lesen 0001 = nur Schreiben 0010 = Lesen und Schreiben Sharing Mode: 000 = Kompatibilitätsmodus 001 = ablehnender Schreib-/Lesemodus 010 = ablehnender Schreibmodus 011 = ablehnender Lesemodus 100 = nicht-ablehnender Modus Inherit Bit: 0 = Zugriff für Unterprozesse erlaubt 1 = Zugriff für Unterprozesse verboten Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst OK	Funktion 44hex (68) DOS 2 IOCTL Block Aktion Steuerdaten an Diskettenstation senden oder von Diskettenstation empfangen Aufruf AH: 68 AL: 4 = Daten senden, 5 = Daten empfangen BL: Laufwerksnummer (0 = aktuelles, 1 = A, 2 = B, etc.) CX: Anzahl der zu sendenden/empfangenden Bytes DS:DX: Zeiger auf Datenpufferbereich Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: AX: Anzahl der übertragenen Bytes
Funktion 3Ehex (62) DOS 2 Close Handle Aktion Datei schließen Aufruf AH: 62 BX: Dateinummer Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst OK	Funktion 44hex (68) DOS 2 IOCTL Status Aktion Gerätestatus für Ein- oder Ausgabe testen Aufruf AH: 68 AL: 6 = Eingabestatus prüfen, 7 = Ausgabestatus prüfen BX: Dateinummer Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: AL: 0 = Gerät ist nicht bereit, 255 = Gerät ist bereit
Funktion 3Fhex (63) DOS 2 Read Handle Aktion Sequentiell aus Datei lesen Aufruf AH: 63 BX: Dateinummer CX: Anzahl Bytes DS:DX: Zeiger auf Datenpuffer Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: AX: Anzahl gelesener Bytes	Funktion 44hex (68) DOS 2 IOCTL Is Changeable Aktion Prüft, ob ein Laufwerk mit auswechselbaren Datenträgern (Disketten) arbeitet oder nicht Aufruf AH: 68 AL: 8 BL: Laufwerk (0 = aktuelles, 1 = A, 2 = B, etc.) Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: AX: 0 = Gerät ist Diskettenstation 1 = Gerät ist Hard- oder RAM-Disk
Funktion 40hex (64) DOS 2 Write Handle Aktion Sequentiell in Datei schreiben Aufruf AH: 64 BX: Dateinummer CX: Anzahl Bytes DS:DX: Zeiger auf Datenpuffer Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: AX: Anzahl geschriebener Bytes	Funktion 45hex (69) DOS 2 Duplicate File Handle Aktion Erzeugt eine zusätzliche Dateinummer für eine Datei Aufruf AH: 69 BX: Dateinummer Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: AX: Neue Dateinummer
Funktion 41hex (65) DOS 2 Delete Directory Entry Aktion Datei im Directory löschen Aufruf AH: 65 DS:DX: Zeiger auf Pfadnamen Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst OK	Funktion 46hex (70) DOS 2 Force Duplicate File Handle Aktion Ordnet die Dateinummer einer zweiten Datei einer anderen Datei zu Aufruf AH: 70 BX: Dateinummer CX: zweite Dateinummer Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst OK
Funktion 42hex (66) DOS 2 Move File Pointer Aktion Dateizeiger weiterbewegen Aufruf AH: 66 BX: Dateinummer AL: Bewegungskode 0: Dateianfang plus Offset in CX 1: aktuelle Zeigerposition plus Offset in CX 2: Dateiende plus Offset in CX (CX kleiner Null !) CX: Offset für Dateizeiger Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: DX:AX: Neue Position des Dateizeigers (32 Bit)	Funktion 47hex (71) DOS 2 Get Current Directory Aktion Ergibt den Pfadnamen des aktuellen Subdirectories Aufruf AH: 71 DL: Laufwerk (0 = aktuelles, 1 = A, etc.) DS:SI: Zeiger auf 64-Byte-Puffer für Pfadnamen Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst OK
Funktion 43hex (67) DOS 2 Get/Set File Attributes Aktion Dateiattribut holen oder setzen Aufruf AH: 67 AL: 0 = Attribute holen, 1 = Attribute setzen CX: (wenn AL = 1) zu setzende Attribute DS:DX: Zeiger auf Pfadnamen Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: CX: (wenn AL = 0 bei Aufruf) Attribute	Funktion 48hex (72) DOS 2 Allocate Memory Aktion Speicher anfordern Aufruf AH: 72 BX: Größe des benötigten Speicherbereichs in Paragraphen Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: BX: Größe des verfügbaren Speicherteils in Paragraphen AX: Segment-Adresse des angeforderten Speicherbereichs

Tabelle 7. Alle MS-DOS-Funktionsaufrufe der Version 3.2 mit Registerbelegung

Ausgefeilte Programmier- technik und praxis- erprobte Musterprogramme in Assembler

Das Intel 16 Bit Assemblerhandbuch bietet Ihnen

- **systematische Programmierkurse** mit sofort nachvollziehbaren Übungsbeispielen, z.B. zur Stringverarbeitung, Interruptprogrammierung oder Makrotechnik;
- **detaillierte Dokumentationen von Architektur und Registern** der Prozessoren 8088 und 80(x)86 sowie der Coprozessoren 80(2)87;
- **den Befehlsvorrat** dieser Prozessoren: übersichtlich aufbereitet und mit ausführlichen Kommentaren;
- **Makrobibliotheken**: u.a. Stringbibliothek;
- **Module eines komfortablen Makroassemblers**: Von Anfang an sind Sie bei der Entwicklung eines Makroassemblers für den 8086-Prozessor dabei. Sie erkennen Aufbau und Funktion der einzelnen Module sowie deren Zusammenwirken. So erhalten Sie Schritt für Schritt ein exzellentes Programmierwerkzeug;
- **bewährte Musterlösungen in Assembler**: Sie erhalten **Druckerutils** wie ein Drucker-Testprogramm, **Schnittstellenprogramme**, u.a. ein Programm zum Datentransfer zwischen PC/MS-DOS-Rechnern, **technische Programme** wie das Muster einer Interruptroutine oder die Programmierung des Timers 8253-5, **mathematische Programme**, u.a. ein Programm CRC-Prüfsummen für Dateien, **Anwenderprogramme** wie Passwörter für Systemdisketten oder eine einfache Schreibmaschine sowie **Utilities** wie zum Directory Update oder Dateien-Dump;
- **Ergänzungsausgaben zum Grundwerk** mit aktuellen Prozessorbeschreibungen (80386), neuen Musterprogrammen, nützlichen Routinen oder auch Erweiterungen der Makrobibliothek.

Mit zusätzlichem Diskettenservice!

Fordern Sie noch heute an:

Intel 16 Bit Assemblerhandbuch

stabiler Ringbuchordner, DIN A 4, ca. 700 Seiten,
Bestell-Nr. 2200, Preis: DM 92,-

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

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 Drucker-codes, Batchdateien, RAM-Disk, Turbo-Pascal-Schnittstelle zu MS-DOS, V.24-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,
Bestell-Nr. 2600, Preis: DM 92,-

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

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.
 - **komplette Softwarepakete** wie das Textverarbeitungsprogramm „SOTEXT“;
 - **anspruchsvolle Anwenderprogramme** wie
 - **Programme für Wirtschaft und Verwaltung**: Netzplantechnik, Terminkalender, Berechnung der Einkommensteuer, Rabattermittlung, Tilgungsplan eines Darlehens, Restwertberechnung...;
 - **mathematische Programme**: Analyse von Funktionsgleichungen (z.B. Kurvendiskussion);
 - **technische Programme**: Messen/Steuern/Regeln, Bestimmen chemischer Formeln...;
 - **Programme für Ausbildung** (z.B. Taschenrechner für komplexe Zahlen), **Privathaushalt** (z.B. Vorratshaltung), **Gesundheitskontrolle** und zur Entspannung **Spiele** (z.B. Zahlenrätsel).
 - **Tools und vielseitige Routinen** für
 - **Grafik**: Datei-Generator „SOGRAF“, Zeichengenerator;
 - **Dateiverwaltung und Dateiorganisation**: Dateigenerator, Komprimieren und Expandieren von Textdateien, Konvertieren von Dateien, Disketten-Editor „SOZAP“, indexsequentielle Dateiorganisation und Reorganisation, Hash-Dateiorganisation; 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 Problembe-schreibung, 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. 1300/70698,
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).

Intel 16 Bit Assemblerhandbuch Programmiertechnik und Programmierung für IBM PCs und Kompatible

Programmierte Unterweisung in Assembler, Trainings-
programm für perfekte Programmierung mit Musterprogrammen
und Routinen, Makroassembler, Makrobibliotheken.



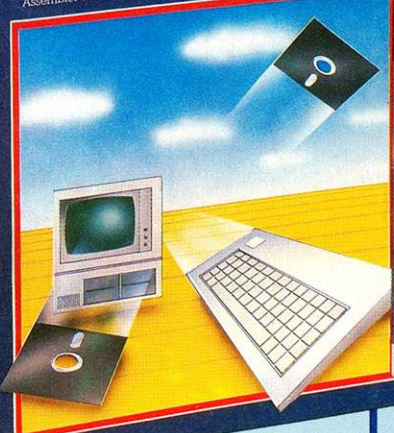
INTEREST VERLAG

Fachverlag
für anspruchsvolle
Freizeitgestaltung

Industriestraße 21
D-8901 Kissing
Tel. 082 33/21 01-0

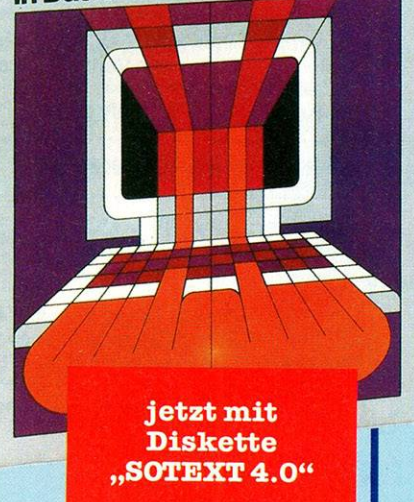
Hard- und Softwaretuning für IBM PCs und Kompatible

- 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 Muster- programme in Basic

Lutz / Schulz-Osterloh
Praxishandbuch für die
erfolgreiche
Programmierung
wirtschaftlicher,
mathematischer und
technischer Programme,
sowie interessanter Spiele



jetzt mit
Diskette
„SOTEXT 4.0“

Funktion 49hex (73) DOS 2 Free Allocated Memory Aktion Speicher freigeben Aufruf AH: 73 ES: Segment-Adresse des freizugebenden Speicherbereiches Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst OK	Funktion 57hex (87) DOS 2 Get/Set Date/Time of File Aktion Datum und Zeit eines Dateieintrags im Directory abfragen oder setzen. Datum und Zeit sind in einem speziellen 16-Bit-Format codiert. Datum: JJJJJJM MMMTTTT (Jahr ab 1980) Zeit: HHHHHMM MMSSSSS Aufruf: AH: 87 AL: 0 = Datum/Zeit lesen 1 = Datum/Zeit setzen BX: Dateinummer CX: bitcodierte Zeitangabe DX: bitcodierte Datumsangabe Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: CX: bitcodierte Zeitangabe DX: bitcodierte Datumsangabe
Funktion 4Ahex (74) DOS 2 Set Block Aktion Größe eines belegten Speicherblocks ändern Aufruf AH: 74 BX: Anzahl Paragraphen ES: Segment-Adresse des zu ändernden Speicherblocks Rückkehr Carry gesetzt: Fehler mit Nummer in AX. Falls Fehler 8 (unzureichender Speicher), dann enthält BX den maximalverfügbaren Speicher.	Funktion 58hex (88) DOS 2 Get/Set Allocation Strategy Aktion Strategie für die Behandlung von Speicheranforderungen festlegen. Möglich sind drei Strategien: 0 = First Fit: Der Speicher wird von niedrigen Adressen her abgesucht. Verwendet wird der erste passende Speicherblock 1 = Best Fit: Der Speicher wird vollständig abgesucht. Verwendet wird der kleinste passende Speicherblock 2 = Last Fit: Der Speicher wird von hohen Adressen her abgesucht. Verwendet wird der oberste passende Speicherblock Die voreingestellte Strategie ist First Fit Aufruf AH: 88 AL: 0 = Strategie holen, 1 = Strategie setzen BX: (wenn AL = 1) First Fit, Best Fit oder Last Fit Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: AX: (wenn AL = 0 bei Aufruf) aktuelle Strategienummer
Funktion 4Bhex (75) DOS 2 Load and Execute Program Aktion Eine Programmdatei wird geladen und ausgeführt Aufruf AH: 75 AL: 0 DS:DX: Zeiger auf Pfadnamen ES:BX: Zeiger auf Parameterblock Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: OK	
Funktion 4Bhex (75) DOS 2 Load Overlay Aktion Eine Overlaydatei wird geladen Aufruf AH: 75 AL: 3 DS:DX: Zeiger auf Pfadnamen ES:BX: Zeiger auf Parameterblock Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: OK	
Funktion 4Chex (76) DOS 2 End Process Aktion Programm beenden und Return-Code zurückgeben Aufruf AH: 76 AL: Return-Code Rückkehr Funktion kehrt nicht zurück	
Funktion 4Dhex (77) DOS 2 Get Return Code of Child Process Aktion Return-Code eines beendeten Tochter-Prozesses lesen Aufruf AH: 77 Rückkehr AX: Return-Code	Funktion 59hex (89) DOS 3 Get Extended Error Code Aktion Erweiterte Informationen über den letzten DOS-Fehler einholen Aufruf AH: 89 BX: 0 Rückkehr AX: erweiterter Fehlercode BH: Fehlerklasse BL: Vorschlag für weiteres Verhalten des Programms CH: ungefähre Ort der Fehlerursache CL, DX, SI, DI, BP, DS, ES: verändert
Funktion 4Ehex (78) DOS 2 Find First File Aktion Ersten Directory-Eintrag zu einem vorgegebenen Dateinamen suchen und zur aktuellen DTA bringen Aufruf AH: 78 CX: Dateiattribut DS:DX: Zeiger auf Pfadnamen Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: OK	Funktion 5Ahex (90) DOS 3 Create Temporary File Aktion Anlegen einer temporären Datei mit eindeutigem Namen Aufruf AH: 90 CX: Dateiattribut DS:DX: Zeiger auf Pfadname plus 13 Bytes für Dateinamen Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: AX: Dateinummer
Funktion 4Fhex (79) DOS 2 Find Next File Aktion Nächsten Directory-Eintrag suchen. Aufruf nur möglich, wenn Funktion 4EH einmal aufgerufen wurde Aufruf AH: 79 Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: OK	Funktion 5Bhex (91) DOS 3 Create New File Aktion Anlegen einer neuen Datei. Fehler, falls bereits eine Datei gleichen Namens existiert Aufruf AH: 91 CX: Dateiattribut DS:DX: Zeiger auf Pfadnamen Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: AX: Dateinummer
Funktion 50hex (80) Reserviert	
Funktion 51hex (81) Reserviert	
Funktion 52hex (82) Reserviert	
Funktion 53hex (83) Reserviert	
Funktion 54hex (84) DOS 2 Get Verify State Aktion Status des Verify-Flags abfragen Aufruf AH: 84 Rückkehr AL: 0 = Verify aus, 1 = Verify ein	Funktion 5Chex (92) bis 61hex (97) Netzwerk-Funktionen
Funktion 55hex (85) Reserviert	Funktion 62hex (98) DOS 3 Get PSP Aktion Adresse des Program Segment Prefix (PSP) für den laufenden Prozeß ermitteln Aufruf AH: 98 Rückkehr BX: Segment-Adresse des aktuellen PSP
Funktion 56hex (86) DOS 2 Change Directory Entry Aktion Directory-Eintrag einer Datei inklusive Zugriffspfad umbenennen Aufruf AH: 86 DS:DX: Zeiger auf alten Pfadnamen ES:DI: Zeiger auf neuen Pfadnamen Rückkehr Carry gesetzt: Fehler mit Nummer in AX, sonst: OK	Funktion 63hex (99) bis 7Fhex (127) Reserviert

Tabelle 7. Alle MS-DOS-Funktionsaufrufe der Version 3.2 mit Registerbelegung


```

[C49] 0: { -----
[5DG] 1: { MSDOS.INC - Include-Datei zum Aufruf von
[C49] 2: { -----
[GNA] 3: { (c) 1987 DOS International
[9TD] 4: { Beschreibung siehe DOS Extra 3, PC-Kurs
[C49] 5: { -----
[000] 6:
[000] 7:
[000] 8:
[D4D] 9: { ----- Globale Vereinbarungen -----
[000] 10:
[VM1] 11: const
[GVF] 12: trap: boolean = false; { Fehlermeld
ung anzeigen }
[E2B] 13: ErrorLevel: byte = 0; { Anzahl Feh
ler }
[N4B] 14: ErrorNum: byte = 0; { Fehlernumm
er }
[04B] 15: ErrorCode: integer = 0; { Fehlercode
}
[H5C] 16: ErrorClass: integer = 0; { Fehlerklas
se }
[P0F] 17: ErrorAction: integer = 0; { vorgeschla
gene Aktion }
[8PE] 18: ErrorLocation: integer = 0; { Bereich de
s Fehlers }
[78C] 19: ErrorMessage: string[40] = 'OK'; { Fehlertext
}
[000] 20:
[000] 21:
[A21] 22: type
[9K5] 23: register = record
[W05] 24: case boolean of
[DB1] 25: true:
[1N9] 26: (ax,bx,cx,dx,bp,si,di,ds,
es,Flags:
integer);
[000] 27: false:
[000] 28: (al,ah,bl,bh,cl,ch,dh,dh:
byte);
[TU0] 30: end;
[000] 31:
[000] 32:
[EM0] 33: var
[3A4] 34: reg: register;
[000] 35:
[000] 36:
[000] 37:
[FWC] 38: { ----- DOS-Funktionen aufrufen -----
[000] 39:
[000] 40:
[000] 41:
[000] 42: procedure dos(nr: byte);
[000] 43:
[000] 44: { MS-DOS-Funktion nr aufrufen }
[000] 45:
[BH1] 46: begin
[M83] 47: reg.ah := nr;
[KV2] 48: MsDos(reg);
[TU0] 49: end;
[000] 50:
[000] 51:
[000] 52:
[YE8] 53: function carry: boolean;
[000] 54:
[VZB] 55: { gibt den Zustand des Carry-Flags aus }
[000] 56:
[BH1] 57: begin
[X37] 58: carry := odd(reg.flags);
[TU0] 59: end;
[000] 60:
[000] 61:
[000] 62:
[B3C] 63: { ----- Fehlerbehandlung -----
[000] 64:
[000] 65:
[000] 66:

```

```

[ETH] 67: procedure GetXErrCode(var code, class, acti
on, loc: integer);
[000] 68:
[65F] 69: { erweiterten Fehlercode melden (nur DOS 3.
x !) }
[04H] 70: { Aufruf nur direkt nach letztem DOS-Aufruf
möglich }
[000] 71:
[BH1] 72: begin
[WN2] 73: reg.bx := 0;
[8P1] 74: dos(89);
[H59] 75: code := reg.ax; { Fehlercode }
[61A] 76: class := reg.bh; { Fehlerklasse }
[N7D] 77: action := reg.bl; { vorgeschlagene Aktio
n }
[3Y7] 78: loc := reg.ch; { Fehlerort }
[TU0] 79: end;
[000] 80:
[000] 81:
[000] 82:
[2T5] 83: procedure error;
[000] 84:
[TWD] 85: { Diese Prozedur kann vom Anwender nach }
[TSE] 86: { seinen Erfordernissen abgeändert werden }
[000] 87:
[VM1] 88: const
[HA1] 89: BEL = #7;
[000] 90:
[EM0] 91: var
[3M4] 92: msg: string[40];
[0E3] 93: nr: integer;
[000] 94:
[BH1] 95: begin
[P43] 96: nr := reg.ax;
[ZOM] 97: GetXErrCode(ErrorCode, ErrorClass, ErrorA
ction, ErrorLocation);
[ZW2] 98: case nr of
[CTB] 99: 1: msg := 'ungültige Funktionsnummer';
[ZP9] 100: 2: msg := 'Datei nicht gefunden';
[W89] 101: 3: msg := 'Pfad nicht gefunden';
[RP9] 102: 4: msg := 'zu viele Dateien offen';
[89A] 103: 5: msg := 'Zugriff nicht möglich';
[AA8] 104: 6: msg := 'Datei nicht offen';
[0EE] 105: 7: msg := 'Speicherkontrollblock beschä
digt';
[MC8] 106: 8: msg := 'zuwenig Speicher';
[REB] 107: 9: msg := 'unkorrekte Segmentadresse';
[4E9] 108: 10: msg := 'ungültige Umgebung';
[ZB9] 109: 11: msg := 'ungültiges Format';
[WRC] 110: 12: msg := 'unzulässiger Zugriffscode';
[KY7] 111: 13: msg := 'ungültige Daten';
[75A] 112: 15: msg := 'ungültiges Laufwerk';
[S1A] 113: 16: msg := 'aktuelles Directory';
[97A] 114: 17: msg := 'nicht das gleiche Gerät';
[GKA] 115: 18: msg := 'keine weiteren Dateien';
[4CC] 116: 19: msg := 'Diskette hat Schreibschutz'
;
[B9D] 117: 20: msg := 'Fehlerhafte Disketteneinhei
t';
[M3A] 118: 21: msg := 'Laufwerk nicht bereit';
[W5E] 119: 22: msg := 'Unzulässiges Disketten-Komm
ando';
[SE6] 120: 23: msg := 'CRC-Fehler';
[WY9] 121: 24: msg := 'Falsche Längenangabe';
[YZ6] 122: 25: msg := 'Suchfehler';
[FN9] 123: 26: msg := 'keine MS-DOS-Diskette';
[9BA] 124: 27: msg := 'Sektor nicht gefunden';
[63A] 125: 28: msg := 'Drucker ohne Papier';
[KN7] 126: 29: msg := 'Schreibfehler';
[Z26] 127: 30: msg := 'Lesefehler';
[UT9] 128: 31: msg := 'Allgemeiner Fehler';
[RK9] 129: 32: msg := 'Sharing-Verletzung';
[5D8] 130: 33: msg := 'Lock-Verletzung';
[8F8] 131: 34: msg := 'Falsche Diskette';
[3UC] 132: 35: msg := 'Zugriff auf FCB nicht mögli
ch';
[V3B] 133: 80: msg := 'Datei existiert bereits';
[FHB] 134: 82: msg := 'Aufgabe nicht durchführbar';
[5SC] 135: 83: msg := 'Fehler in der Fehlerbehandl
ung';
[G8A] 136: else msg := 'Unbekannter Fehler';
[293] 137: end; { case }
[BW4] 138: if not trap then
[BH1] 139: begin
[9Z2] 140: writeln;
[E3B] 141: writeln(BEL, '+++++ Fehler', nr:3, ':
', msg);

```



```

[BM7] 142:      repeat until KeyPressed;
[TU0] 143:      end;
[GV7] 144:      ErrorLevel := ErrorLevel + 1;
[PX4] 145:      ErrorNum := nr;
[R65] 146:      ErrorMessage := msg;
[TU0] 147: end;
[000] 148:
[000] 149:
[YX6] 150: procedure ClearError;
[000] 151:
[GG9] 152: { Fehlerstatus zurücksetzen }
[000] 153:
[BH1] 154: begin
[H6D] 155:   trap := false;      { Fehlermeldung anzeigen }
[WC9] 156:   ErrorLevel := 0;    { Anzahl Fehler }
[XL9] 157:   ErrorNum := 0;       { Fehlernummer }
[VW8] 158:   ErrorCode := 0;    { Fehlercode }
[WH9] 159:   ErrorClass := 0;   { Fehlerklasse }
[ZSC] 160:   ErrorAction := 0; { vorgeschlagene Aktion }
[56C] 161:   on
[4C9] 162:   ErrorLocation := 0; { Bereich des Fehlers }
[TU0] 163: end;
[000] 164:
[000] 165:
[WNB] 166: { ---- Ende Bibliothek MSDOS.INC -----
          ----- }

```

Listing 1. Eine Include-Datei zum Aufruf von MS-DOS-Funktionen unter Turbo Pascal

```

[VMC] 0: program xdir; { (c) 1988 DOS International }
[000] 1:
[KKE] 2: { Directory mit Datei-Attributen auflisten }
[18E] 3: { angezeigt werden auch versteckte und }
[YGE] 4: { System-Files sowie Volume und Directories }
[000] 5:
[000] 6:
[9W4] 7: {$I msdos.inc }
[000] 8:
[000] 9:
[D56] 10: const { Attribute }
[164] 11:   _ReadWrite = 0;
[DP3] 12:   _ReadOnly = 1;
[D83] 13:   _Hidden = 2;
[YE3] 14:   _System = 4;
[LD3] 15:   _Volume = 8;
[AA5] 16:   _SubDirectory = 16;
[XW3] 17:   _Archive = 32;
[000] 18:
[000] 19:
[A21] 20: type
[RP6] 21:   PfadName = string[40];
[SR5] 22:   pointer = ^byte;
[866] 23:   DirEntryType = record
[DR9] 24:     reserviert: array[0..20]
       of byte;
[WC5] 25:     Attribut: byte;
[L44] 26:     Zeit: integer;
[PR4] 27:     Datum: integer;
[KH5] 28:     LaengeLo: integer;
[PG5] 29:     LaengeHi: integer;
[WH6] 30:     Name: array[1..13] of ch
       ar;
[TU0] 31:   end;
[000] 32:
[EM0] 33: var
[F48] 34:   DirEntry: ^DirEntryType;
[CR2] 35:   attr: byte;
[B43] 36:   x: integer;
[8K2] 37:   y: real;
[3X5] 38:   Datei, Pfad: PfadName;
[000] 39:
[000] 40:

```

```

[000] 41:
[MZB] 42: procedure GetDTA(var Adresse: pointer);
[000] 43:
[BNA] 44: { Disk Transfer Adresse lesen }
[000] 45:
[BH1] 46: begin
[8N1] 47:   dos(47);
[B19] 48:   Adresse := ptr(reg.es, reg.bx);
[TU0] 49: end;
[000] 50:
[000] 51:
[000] 52:
[G3M] 53: function FirstFile(Name: PfadName; Attribut
       : integer): PfadName;
[000] 54:
[DKB] 55: { ersten Directory-Eintrag suchen }
[000] 56:
[VM1] 57: const
[XB1] 58:   NUL = #0;
[000] 59:
[EM0] 60: var
[MK3] 61:   DTA: pointer;
[XG3] 62:   fn: PfadName;
[000] 63:
[BH1] 64: begin
[WF2] 65:   GetDTA(DTA);
[XW3] 66:   Name := Name + NUL;
[Y46] 67:   reg.ds := seg(Name[1]);
[G76] 68:   reg.dx := ofs(Name[1]);
[ND5] 69:   reg.cx := Attribut;
[XN1] 70:   dos(78);
[ZV5] 71:   if carry then error;
[TU6] 72:   DirEntry := addr(DTA^);
[1T5] 73:   fn := DirEntry^.Name;
[2F7] 74:   fn[0] := chr(pos(NUL, fn) - 1);
[EU4] 75:   FirstFile := fn;
[TU0] 76: end;
[000] 77:
[000] 78:
[000] 79:
[000] 80:
[AU8] 81: function NextFile: PfadName;
[000] 82:
[TXB] 83: { nächsten Directory-Eintrag suchen }
[000] 84:
[VM1] 85: const
[XB1] 86:   NUL = #0;
[000] 87:
[EM0] 88: var
[MK3] 89:   DTA: pointer;
[XG3] 90:   fn: PfadName;
[000] 91:
[BH1] 92: begin
[WF2] 93:   GetDTA(DTA);
[3P1] 94:   dos(79);
[ZV5] 95:   if carry then error;
[TU6] 96:   DirEntry := addr(DTA^);
[1T5] 97:   fn := DirEntry^.Name;
[2F7] 98:   fn[0] := chr(pos(NUL, fn) - 1);
[KS4] 99:   NextFile := fn;
[TU0] 100: end;
[000] 101:
[000] 102:
[CW8] 103: procedure dir(Name: PfadName);
[BH1] 104: begin
[9Z2] 105:   writeln;
[MA3] 106:   trap := true;
[664] 107:   ErrorLevel := 0;
[000] 108:
[K98] 109:   { Volume-Namen bestimmen }
[000] 110:
[N8A] 111:   Datei := FirstFile(Name, _Volume);
[BT6] 112:   if ErrorLevel > 0 then
[BH1] 113:     begin
[ZM9] 114:       Datei := '(kein Name vorhanden)';
[664] 115:       ErrorLevel := 0;
[DL0] 116:     end
[AHA] 117:   else delete(Datei, pos('.', Datei), 1);
[000] 118:
[R78] 119:   writeln('Volume: ', Datei);
[9Z2] 120:   writeln;
[000] 121:
[W77] 122:   { erste Datei suchen }
[000] 123:
[Z1K] 124:   Datei := FirstFile(Name, _SubDirectory +
       _Hidden + _System);
[BT6] 125:   if ErrorLevel > 0 then
[BH1] 126:     begin
[OLA] 127:       writeln('*** Datei nicht gefunden');

```



```

[DL0] 128: end
[Al1] 129: else
[XRH] 130: writeln ('Dateiname Attribute Datu
m Zeit Länge');
[ZM3] 131: writeln ('');
[3F8] 132: repeat { Dateien anzeigen }
[FB3] 133: write(Datei);
[B25] 134: GotoXY(14, WhereY);
[U58] 135: attr := DirEntry^.Attribut;
[825] 136: GotoXY(16,WhereY);
[71A] 137: if (attr and 32) <> 0 then { Archive
}
[1A2] 138: write ('A')
[Al1] 139: else
[7G2] 140: write ('-');
[12B] 141: if (attr and 16) <> 0 then { Sub-Dire
ctory }
[7G2] 142: write('D')
[Al1] 143: else
[7G2] 144: write ('-');
[AS9] 145: if (attr and 4) <> 0 then { System }
[7G2] 146: write('S')
[Al1] 147: else
[7G2] 148: write ('-');
[ZH9] 149: if (attr and 2) <> 0 then { Hidden }
[7G2] 150: write('H')
[Al1] 151: else
[7G2] 152: write ('-');
[BGA] 153: if (attr and 1) <> 0 then { Read Onl
y }
[7G2] 154: write('R')
[Al1] 155: else
[7G2] 156: write ('-');
[CS5] 157: x := DirEntry^.Datum;
[525] 158: GotoXY (24,WhereY);
[YR8] 159: write ((x AND 31):2,'. ');
{ Tag
[CVA] 160: write (((x AND 480) DIV 32):2,'. ');
{ Monat
[WOB] 161: write (((Hi(x) AND $FE) DIV 2)+80):2
); { Jahr
[885] 162: x := DirEntry^.Zeit;
[625] 163: GotoXY (34,WhereY);
[A1B] 164: write (((Hi(x) AND $F8) DIV 8):2,'. ');
{ Stunde
[FVA] 165: write (((x AND $7E0) DIV 32):2,'. ');
{ Minute
[R2A] 166: write (((x AND 31)*2):2);
{ Sekunde
[4UG] 167: y := (DirEntry^.LaengeLo AND $7FFF)
+ 65536.0 * DirEntry^.LaengeHi;
[AlA] 168: if (DirEntry^.LaengeLo AND $8000) <>
0 then
[443] 169: y := y + 32768.0;
[725] 170: GotoXY (44,WhereY);
[EE3] 171: write (y:10:0);
[922] 172: writeln;
[6Y4] 173: Date1 := NextFile;
[9U5] 174: until ErrorLevel > 0;
[TU0] 175: end;
[000] 176:
[BH1] 177: begin
[X52] 178: repeat
[MPE] 179: write('Bitte Suchpfad eingeben (z.B. a:
\unter\*.*) : ');
[PE3] 180: readln(Pfad);
[TA7] 181: if Pfad <> '' then dir(Pfad);
[922] 182: writeln;
[NL3] 183: until Pfad = '';
[5T0] 184: end.

```

Listing 2. Anzeige des Directories als Anwendung von Listing 1

```

[F3A] 0: { Die Verwendung von "Command.com" }
[8KA] 1: { parallel zu einem Turbo Pascal }
[1P3] 2: { Programm }
[000] 3:
[14E] 4: { Übersetzen als COM-Datei mit Stackgröße =
40 }

```

```

[000] 5:
[0S6] 6: PROGRAM CommandEscape;
[000] 7:
[14H] 8: CONST Trigger = '!'; { Dieses Zeichen ru
ft "Command.com" }
[FWF] 9: Exit = #02; { Dieses Zeichen be
endet das Programm }
[000] 10:
[PZ3] 11: VAR ShellStr,
[TH6] 12: CurrComm : STRING[255];
[KE3] 13: i,j : INTEGER;
[FR2] 14: c : CHAR;
[VXE] 15: Regs : RECORD ax,bx,cx,dx,bp,si
,di,ds,es,fl: INTEGER;
[7MH] 16: END;
[000] 17:
[000] 18:
[4U6] 19: { DOS-Befehl ausführen }
[000] 20:
[RB5] 21: PROCEDURE DoCommand;
[BV7] 22: VAR Block: ARRAY[0..7] OF INTEGER;
[B21] 23: BEGIN
[000] 24:
[7HH] 25: { Kein Befehl -> "command" starten und au
f Eingabe warten }
[M14] 26: CurrComm := #13;
[KTD] 27: CurrComm := Chr(Length(CurrComm)-1)+CurrC
omm;
[000] 28:
[Y19] 29: { Aufbau des Parameterblocks:
[V2G] 30: 1. Wort: 0 = Umgebung von diesem
Programm kopieren
[HK9] 31: 2. Wort: Offset des Kommandos
[4GA] 32: 3. Wort: Segment des Kommandos
[VDF] 33: 4. bis 7. Wort: für Dateien, hier nicht
gebraucht }
[TC3] 34: Block[0] := 0;
[FW7] 35: Block[1] := Ofs(CurrComm)+1;
[HF7] 36: Block[2] := Seg(CurrComm);
[000] 37:
[8FE] 38: Regs.ax := $4B00; { AX = Funkt
ionsnummer
[MW7] 39: Regs.dx := Ofs(ShellStr)+1;
[BYH] 40: Regs.ds := Seg(ShellStr); { DS:DX = Ad
resse des Dateinamen
[V37] 41: Regs.bx := Ofs(Block[0]);
[XWG] 42: Regs.es := Seg(Block); { ES:BX = Ad
resse des Parameters
[7Y2] 43: MsDos(Regs);
[7M0] 44: END;
[000] 45:
[000] 46:
[B21] 47: BEGIN
[4KA] 48: { Genauer Pfad für "command.com" }
[RV9] 49: ShellStr := 'c:\command.com'+Chr(0);
[000] 50:
[W75] 51: { Unser Zähler }
[8D1] 52: i := 0;
[000] 53:
[2W3] 54: WHILE TRUE DO BEGIN
[UGC] 55: i := i+1; { Zähler inkrementier
en ... }
[KY2] 56: GotoXY (1,1);
[34A] 57: Write (i); { ... und ausgeben
}
[000] 58:
[TKF] 59: { Die Achtleife gibt dem Benutzer eine b
essere Chance,
[7VC] 60: mit der Eingabe durchzukommen!
}
[7K3] 61: FOR j := 1 TO 50 DO
[3F6] 62: IF KeyPressed THEN BEGIN
[NF2] 63: ClrScr;
[WR2] 64: Read (KBD,c);
[KG4] 65: IF c=Trigger THEN
[1V2] 66: DoCommand
[RG4] 67: ELSE IF c=Exit THEN
[TT1] 68: Halt (0);
[7M0] 69: END;
[7M0] 70: END;
[5LO] 71: END.

```

Listing 3. MS-DOS-Kommandos ausführen mit Turbo Pascal 3.0



bietet selbstverständlich den schnellsten Weg, Zeichen auszugeben oder Eigenschaften der Anzeige zu ändern.

Jetzt ein kurzes Wort zur Hardware. Da er nicht programmierbar ist, interessiert uns der Monitor hier wenig. Wichtig ist er aber schon – ein schlechter Monitor kann durch trübe, häßliche Farben oder eine unscharfe Anzeige den Spaß am Computer total verderben. Die anderen Hardware-Teile sind fast immer zusammen auf einer Platine zu finden. Die Platine wird im allgemeinen »Bildschirmadapter« oder »Grafikkarte« genannt. Sie steckt entweder in einem Erweiterungs-Slot oder (in neueren PCs) bereits auf der Hauptplatine.

Es gibt fünf wichtige Grafikkarten für den PC, die verschiedene Anwendungsbereiche abdecken

Die ersten PCs wurden nur mit dem »Monochrome Display Adapter«, auf deutsch etwa »Karte zur einfarbigen Bildschirmsteuerung«, kurz MDA, ausgeliefert. Diese Karte bietet ein sehr schönes Schriftbild auf einem 80 (Spalten) mal 25 (Zeilen) – Bildschirm, ist aber von Haus aus weder farb- noch grafikfähig. Diese Einschränkungen führten schnell zu einem Nachfolger, der Farbgrafikkarte CGA (»Color Graphic Adapter«). Die CGA-Karte bietet, wie die MDA-Karte, einen 80 mal 25 Text-Bildschirm, aber die Schrift ist lange nicht so schön wie beim Vorgänger. Warum würde man dann eine CGA-Karte anschaffen? Die Antwort ist zweiteilig: Grafik und Farben. Außer einem Text-Modus in 16 Farben bietet die CGA-Karte zwei Grafik-Modi, 320 (waagrecht) x 200 (senkrecht) Punkte in vier Farben, oder 640 x 200 Punkte in zwei Farben. Zusätzlich hat die CGA-Karte einen neuen Text-Modus, 25 Zeilen mit je 40 (doppelgroßen) Zeichen, auch in 16 Farben.

Das nächste Mitglied der Familie ist das einzige, das nicht ursprünglich von IBM stammt. Da die Anwender im Geschäft eindeutig Grafik wollten, aber auf die gestochene scharfe Schrift der MDA-Karte nicht verzichten wollten, erfand die amerikanische Firma Hercules die MGA-Karte (»Monochrome Graphic Adapter«). Diese Karte bietet die zwei großen Eigenschaften der beiden anderen Karten, nämlich das schöne Schriftbild und die Grafik, aber ohne Farben. Wenn die CGA-Karte auch die Standardkarte geworden ist, ist die MGA-Karte (auch Hercules-Karte genannt) kaum weniger populär. Die Grafik-Auflösung dieser Karte ist sogar besser als die der CGA-Karte, 720 (waagrecht)

Der PC – glasklar

Teil (4)

In dieser vierten Folge unseres Kurses geht es darum, wie man den Bildschirm steuert. Dieses Thema mag einfach klingen, aber ein kurzer Blick auf die Vielfalt der verschiedenen Grafikkarten und Bildschirm-Modi führt schnell zur Verwirrung. Auf den folgenden Seiten werden wir Licht in diese dunkle Ecke bringen, so daß jeder PC-Anwender den Bildschirm schnell und bequem manipulieren kann.

Zuerst lernen Sie die verschiedenen Mitglieder der Familie der Bildschirmadapter kennen. Die Tatsache, daß es mehrere Mitglieder dieser Familie gibt, ist historisch bedingt. Am Anfang (das ist erst sieben Jahre her) meinte IBM, man würde den PC nur für geschäftliche Aufgaben einsetzen. Deshalb würde man keine Grafik benötigen, und so wurden die ersten PCs ausschließlich mit dem nicht-grafikfähigen Monochrom-Adapter ausgeliefert. Nach einiger Zeit zeigte es sich jedoch, daß IBM sich geirrt hat, und die anderen Mitglieder der Grafikkartenfamilie wurden erfunden, um die steigenden Ansprüche der Anwender auf immer bessere Grafiken zu erfüllen. Am weitesten verbreitet ist dabei der sogenannte »Color Graphic Adapter« (CGA). Diese Farbgrafikkarte gehört inzwischen zur Grundausstattung der meisten PCs.

Die gesamte Bildschirmsteuerung beim PC läßt sich in sechs Komponenten aufteilen: Ganz unten steht der Monitor, der die Aufgabe hat, elektrische Signale in Licht umzuwandeln. Die nächsten zwei Kompo-

nenten über dem Monitor, die Bildschirmsteuerlogik und der Bildschirmspeicher, sind die anderen Hardware-Teile der Bildschirmsteuerung. Die Bildschirmsteuerlogik nimmt digitale Informationen aus dem Bildschirmspeicher heraus, und wandelt diese in elektrische Signale für den Monitor um. Im Bildschirmspeicher werden diese Informationen abgelegt – wie und in welcher Form wird uns später noch sehr beschäftigen. Alle drei Hardware-Teile sind notwendig, um ein Bild zu erzeugen.

Die restlichen drei Komponenten der Bildschirmsteuerung sind Software-Komponenten, die die Hardware programmieren, um die gewünschte Anzeige zu erzeugen: BIOS, MS-DOS und schließlich das Anwender-Programm. Das Anwender-Programm spricht die Bildschirm-Hardware normalerweise über das Betriebssystem oder über das BIOS an. Es ist aber durchaus möglich, daß das Anwender-Programm die Bildschirm-Hardware direkt manipuliert. Zum Beispiel werden wir an einigen Stellen vorführen, wie man mit dem Bildschirmspeicher umgehen kann. Dies

mal 348 (senkrecht) Punkte. Jedoch ist die Einfarbigkeit der MGA-Karte für viele Anwendungen ein bedeutender Minuspunkt. Im Gegensatz zur CGA-Karte wird die Hercules-Karte im Grafikmodus nicht vom BIOS unterstützt, so daß Besitzer von Hercules-Karten häufig Schwierigkeiten bei Grafik-Programmen haben.

In der letzten Zeit hat sich noch eine Bildschirmkarte auf dem Markt durchsetzen können. Die EGA-Karte (»Enhanced Graphics Adapter«) hat ein Schriftbild, das kaum von dem der MDA-Karte zu unterscheiden ist und bietet eine Grafik mit 640 (waagrecht) mal 350 (senkrecht) Punkten in 16 Farben. Diese 16 Farben können gleichzeitig verwendet werden; zur Wahl steht eine Palette von 64 Farben. Außerdem kann die EGA-Karte (zumindest nach dem IBM-Vorbild) die MDA- und CGA-Karten emulieren. Inzwischen gibt es auch erweiterte EGA-Karten von anderen Herstellern, die zusätzlich eine MGA-Emulation, bessere Auflösung (normalerweise 640 mal 480 Punkte), und/oder mehrere Farben (bis 64 gleichzeitig aus einer Palette von 256) anbieten. Es ist nun fast soweit, daß die EGA-Karte die neue Standardkarte geworden ist.

Wie üblich in der PC-Welt, bevor der neue Standard ganz feststeht, gibt es etwas besseres. Das bessere heißt in diesem Fall »VGA« (Video Gate Array) und ist mit der PS/2-Reihe von IBM auf der Welt erschienen. In der Zwischenzeit gibt es aber auch VGA-Karten für die »alten« PCs und ATs. Diese Karte bietet zum Beispiel eine Auflösung bis 640 (waagrecht) x 480 (senkrecht) mit 16 Farben gleichzeitig aus einer Palette von 262144. Auf die Frage: »Was kann man mit mehr als einer Viertelmillion Farben machen?«, gibt es wirklich nur eine Antwort: »Sehr, sehr schöne Bilder!« Ein weiterer Modus der VGA-Karte bietet 256 Farben gleichzeitig, allerdings mit einer relativ niedrigen Auflösung von 320 x 200. Dies sind die Merkmale der ursprünglichen VGA-Karte; Sie dürfen nicht überrascht sein, wenn es kompatible Karten von anderen Herstellern gibt, die wesentlich mehr bieten (zum Beispiel, 800 (waagrecht) x 600 (senkrecht) Punkte in 256 aus 262144 Farben). Die Entwicklung der Grafikkarten läßt sich nur bewundern.

Die Bildschirmsteuerung über das BIOS ist in den meisten Programmiersprachen problemlos

Wie Sie bereits wissen, werden alle BIOS-Routinen über sogenannte »Software-Interrupts« aufgerufen. In-

Nr. (AH)	Funktion	Register	Eingabe	Ausgabe
0	Bildschirmmodus setzen		AL: Modus	
1	Cursorgröße setzen		CH: Anfangszeile CL: Endzeile	
2	Cursor-Adresse setzen		BH: Bildschirmseite DH: Zeile DL: Spalte	
3	Cursor-Adresse abfragen		BH: Bildschirmseite	DH: Zeile DL: Spalte
4	Lichtgriffeladresse holen			AH: Auslösung (0..1) DH: Zeichenzeile DL: Zeichenspalte CH: Pixelzeile BX: Pixelspalte
5	Bildschirmseite setzen		BH: Bildschirmseite	
6	Bildschirm hochrollen		AL: Anzahl Zeilen DH: untere Zeile CH: obere Zeile CL: linke Spalte DL: rechte Spalte BH: Leerzeilen-Attribut	
7	Bildschirm nach unten rollen		AL: Anzahl Zeilen DH: untere Zeile CH: obere Zeile CL: linke Spalte DL: rechte Spalte BH: Leerzeilen-Attribut	
8	Zeichen an Cursorposition lesen		BH: Bildschirmseite	AL: Zeichen AH: Attribut
9	Zeichen und Attribut schreiben		BH: Bildschirmseite BL: Attribut AL: Zeichen CX: Anzahl	
10	Nur Zeichen schreiben		BH: Bildschirmseite AL: Zeichen CX: Anzahl	
11	Farbpalette setzen		BL: Farbe (nur Grafikmodus) BH: 0 oder 1 BL: IRGB-Bits (falls BH = 0) BL: Palette (falls BH = 1)	
12	Grafikpunkt setzen		DX: Pixelzeile CX: Pixelspalte AL: Farbe	
13	Grafikpunkt lesen		DX: Pixelzeile CX: Pixelspalte	AL: Pixelfarbe
14	Zeichen in TTY-Modus schreiben		AL: Zeichen BL: Farbe (nur Grafikmodus)	
15	Bildschirmparameter holen			BH: Bildschirmseite AH: Anzahl Textspalten AL: Bildschirmmodus

Tabelle 1. Unterfunktionen des BIOS-Interrupts 16

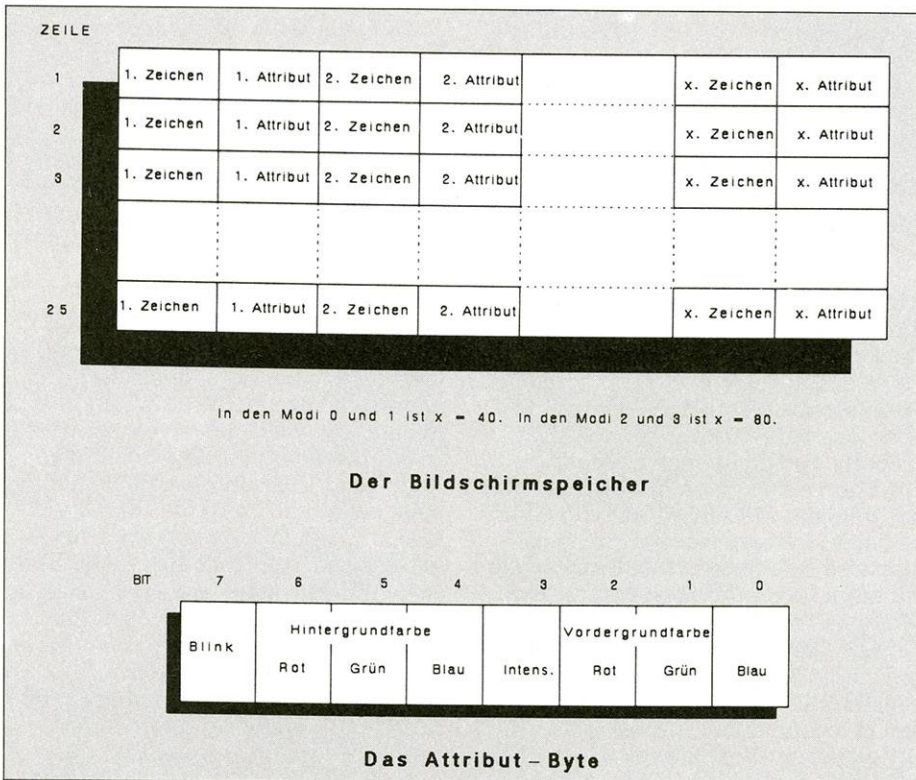


Bild 1. Der Aufbau des Bildschirmspeichers im Text-Modus

Anfangsadresse		Endadresse
B8000	Seite 0	B87CF
B8800	Seite 1	B8FCF
B9000	Seite 2	B97CF
B9800	Seite 3	B9FCF
BA000	Seite 4	BA7CF
BA800	Seite 5	BAFCF
BB000	Seite 6	BB7CF
BB800	Seite 7	BBFCF

$$\text{Anfang} = \text{B8000} + \text{Seite} \times 800$$

$$\text{Ende} = \text{Anfang} + 7\text{CF}$$

0	Zeile 1	4F
50	Zeile 2	9F
A0	Zeile 3	EF
F0	Zeile 4	13F
140	Zeile 5	18F
780	Zeile 25	7CF

$$\text{Anfang} = (\text{Zeile} - 1) \times 50$$

$$\text{Ende} = \text{Anfang} + 4\text{F}$$

Bild 2. Zeilen und Seiten in den Text-Modi 0 und 1

errupt 16 wird verwendet, um die BIOS-Routinen für die Bildschirmsteuerung aufzurufen.

Für den Bildschirm gibt es 16 BIOS-Funktionen (beim PC/AT mehr), die über Interrupt 16 zu erreichen sind (Tabelle 1). Um die gewünschte Funktion auszuwählen, muß man das AH-Register mit einer Zahl von 0 bis 15 laden, bevor der Interrupt ausgelöst wird, wie in der Tabelle zu sehen ist. Manche Funktionen brauchen zusätzliche Parameter, die ebenfalls in Tabelle 1 aufgeführt sind.

In Tabelle 2 sehen Sie gleich, daß die meisten BIOS-Funktionen bereits in den Hochsprachen Turbo Pascal und GW-Basic zur Verfügung stehen. Dies bedeutet, daß Sie die BIOS-Funktionen nur direkt ansprechen müssen,

wenn die Hochsprache die gewünschte Funktion nicht anbietet. Dies ist zum Beispiel der Fall mit Turbo Pascal 3.0 und der Funktion 5. Natürlich können Sie alle Funktionen auch dann direkt einsetzen, wenn sie in der Hochsprache angeboten werden. Diese Vorgehensweise ist häufig sehr vorteilhaft, weil die BIOS-Funktionen vielfach schneller als ihre Gegenstücke in einer Hochsprache sind. Alle Bildschirmadapter lassen sich in mehreren Modi betreiben. Der Betriebsmodus legt drei Eigenschaften der Bildschirmanzeige fest: Es gibt reine Text-Modi, womit keine Grafik dargestellt werden kann und auch Grafik-Modi, die sowohl Text als auch Grafik auf den Bildschirm bringen können. Der jeweils gewählte

Modus bestimmt auch, wie viele Farben gleichzeitig gezeigt werden können. Auch die effektive Grafikauflösung wird durch die Festlegung eines Bildschirmmodus gesteuert. In Tabelle 3 sind die verschiedenen Modi der Bildschirmdkarten zusammengefaßt. Die Nummer in der linken Spalte wird als Kennzeichen für den Modus verwendet, man spricht vom »Modus 6« oder »Modus 1«. Die Spalten Art, Auflösung und Farben beschreiben kurz den Modus, und rechts in der Tabelle wird angegeben, welche Karte(n) den Modus anbietet(en).

Im folgenden erfahren Sie Grundlegendes über die Modi der CGA-Karte, da fast alle PCs entweder mit einer echten CGA-Karte oder mit einer Emulation ausgerüstet sind. Die MGA- oder EGA-Anwender können die hier beschriebenen oder ähnliche BIOS-Routinen (in den entsprechenden Modi) auch einsetzen und sollten auch die MGA- und EGA-Routinen besser verstehen und einsetzen können, nachdem die Eigenschaften der CGA-Karte verständlich geworden sind. Wie man in der Tabelle sieht, sind die Betriebsarten der CGA-Karte die Modi 0 bis 6. Die Nummer 7 ist der Modus der MDA-Karte, und höhere Zahlen sind nur mit einer EGA-Karte möglich. Die fehlenden Zahlen 8 und 9 sind Modi des PC-Junior, der völlig uninteressant ist, und die 11 und 12 sind interne Modi der EGA-Karte. Die Hercules-Karte hat keinen offiziell zugeordneten Modus.

Wer die Tabelle etwas näher betrachtet, der bemerkt, daß einige Modi sich sehr ähnlich sind. Es gibt drei Paare, 0 und 1, 2 und 3, und 4 und 5, die sich nur in der Wiedergabe von Farben unterscheiden. Wieso ist das der Fall? Die Antwort liegt an der Hardware, in diesem Fall an der Grafikkarte und dem Monitor. Als die CGA-Karte erstmals auf den Markt kam, waren Farbmonitore drei- bis fünfmal so teuer, wie sie es heute sind. Deshalb hat die originale CGA-Karte zwei Monitor-Ausgänge, einen für einen Farbmonitor und einen für einen Monochrom-Monitor. Wer Grafik wollte, aber die paar Tausender für einen Farbmonitor nicht hinblättern konnte, wurde damit auch bedient. Er konnte nämlich einen günstigen Monochrom-Monitor an die Farbgrafikkarte anschließen, und damit die Grafikfähigkeit der CGA-Karte ausnutzen, allerdings ohne Farbe. Dies ging aber nur dann, wenn die CGA-Karte kein Farbsignal an den Monochrom-Monitor weitergab, und dies ist der Grund, daß die drei farblosen Modi bestehen. Die drei Modi gestatten eine Zusammenarbeit zwischen der Farbgrafikkarte und einem Monochrom-Monitor. Wenn Sie mit einem Monochrom-Monitor arbeiten,

Nr.	BIOS-Routine	Turbo Pascal	GW-Basic
0	Modus setzen	TextMode GraphMode GraphColorMode HiRes	Screen, Width
1	Cursorgöße setzen	—	Locate
2	Cursorposition setzen	GotoXY	Locate
3	Cursorposition abfragen	WhereX WhereY	Csrlin Pos
4	Lichtgriffelposition abfragen	—	—
5	aktive Seite setzen	—	Screen
6	Fenster nach oben rollen	—	—
7	Fenster nach unten rollen	—	—
8	Zeichen lesen	—	Screen
9	Zeichen und Attribut schreiben	Write TextColor TextBackground	Print Color
10	Zeichen schreiben	Write	Print
11	Farbpalette auswählen	Palette GraphBackground	Color
12	Pixel schreiben	Plot	PSet PReset
13	Pixel lesen	GetDotColor	Point
14	TTY-Zeichen schreiben	Write TextColor	Print Color
15	Modus abfragen	—	—

Tabelle 2. Die BIOS-, Basic- und Pascal-Routinen der Bildschirmsteuerung

und ein komisches Bild bekommen, überprüfen Sie, ob Modus 1, 3 oder 4 verwendet wurde. Wenn dies der Fall ist, stellen Sie entsprechend auf Modus 0, 2 oder 5 um, um ein besseres Ergebnis zu erzielen. Das Gegenteil gilt für diejenigen, die einen Farbmonitor verwenden. Falls die Farben plötzlich verschwinden (oder, mit einigen CGA-Karten, fest auf drei unerwünschten Farben bleiben, auch wenn Sie andere Farben programmiert haben), prüfen Sie, ob versehentlich Modus 0, 2 oder 5 selektiert wurde. Wenn das der Fall sein sollte, so steigen Sie auf den entsprechenden Modus 1, 3 oder 4 um.

Die CGA-Karte kann 16 Farben darstellen. Diese Anzahl kommt nicht von ungefähr, sondern ist eine Konsequenz der Arbeitsweise dieser Karte. Das Signal zum Monitor wird aus vier Komponenten zusammengesetzt, wobei jede Komponente entweder ins Signal eingemischt wird, oder

aus dem Signal ausgelassen wird. Vier Komponenten mit je zwei möglichen Zuständen ergeben 2x2x2x2 ist gleich 16 komplette Signale, die jeweils eine Farbe wiedergeben. Drei von den vier Komponenten sind die Farbkomponenten Rot, Blau und Grün. Die vierte Komponente heißt »Intensität«, die etwa als Helligkeit zu verstehen ist. Wenn die Intensitäts-Komponente aktiv ist, bekommt man einfach eine hellere Version der gleichen Grundfarbe (Tabelle 4). Wie die Farben tatsächlich aussehen, hängt sehr stark vom Monitor ab.

Manche Monitore können überhaupt nichts mit Intensität anfangen, das heißt, die dunkle und die helle Version einer Farbe sehen gleich aus, während andere Monitore Schwierigkeiten mit manchen Farben (typischerweise Braun, Gelb und Grau) aufweisen. Wie es im Einzelfall bei Ihnen aussieht, können Sie nur durch Ausprobieren feststellen.

Tabelle 5 zeigt die Funktionen der Sprachen Turbo Pascal und GW-Basic, die den Bildschirmmodus setzen. Hier ist kein Zugriff auf das BIOS oder die Hardware notwendig, da alle Modi bequem und einfach von der Hochsprache aus gesetzt werden können. Jedoch bieten weder Pascal noch Basic eine einfache Methode, um den Modus abzufragen. Hier könnte man die BIOS-Routine »Modus abfragen« direkt aufrufen, jedoch gibt es eine noch einfachere Möglichkeit. Das BIOS speichert die Nummer des Modus an einer festen Adresse, nämlich 449hex. Mit der Pascal-Zeile

```
Modus := Mem [$40:$49];
```

oder mit den Basic-Zeilen

```
DEF SEG = &H40
```

```
Modus = PEEK (&H49)
```

```
DEF SEG
```

bekommt man den aktuellen Modus in der Variablen »Modus« zurück. Der Bildschirmspeicher wird in zwei Weisen verwendet. In den Text-Modi werden ASCII-Codes in den Bildschirmspeicher geschrieben, und die CGA-Karte wandelt diese Codes in Punktmuster um, die wiederum auf dem Bildschirm gezeigt werden. In den Grafik-Modi werden Punkte in Form verschlüsselter Farben in den Speicher geschrieben. Die CGA-Karte mischt das Farbsignal nach dem Speicherinhalt zusammen, muß aber keine weitere Umwandlung vornehmen.

Die CGA-Karte kann Text in 16 Farben darstellen, während die Grafik weit weniger farbenfroh ist

Der Bildschirmspeicher ist ein reservierter Bereich im Gesamtspeicher des Prozessors, der sowohl vom Prozessor (sprich: vom Programm) als auch von der Bildschirmsteuerlogik gelesen und geschrieben werden kann. Dieser Speicher ist physika-

ALBATROS-COMPUTER GmbH

EIN NAME DEN MAN SICH MERKEN SOLLTE

7107 Bad Wimpfen · Magnusstr. 9
Tel. (0 70 63) 70 57 (kein Ladenverkauf)

Computer aus eigener Fertigung, unser Service hört nicht beim Computerzubehör auf.
Wir bieten für Mess- und Betriebsdatenerfassung die gesamte Sensorik, auch in Einzel- und Sonderanfertigung.

Als Einführung bieten wir:

Komplettes Textverarbeitungssystem mit Textmaker:

1. Computer 100% kompatibel
 - 4,77/8 MHz schaltbar 8 Slots
 - 512 KB RAM
 - 2 Diskettenlaufwerke 360 KB
 - Videokarte
 - Centronics Port
2. Monitor Monochrom
3. Matrix Drucker Oki Microline 182
 - mit deutschem Handbuch oder gleichwertiges ähnliches Gerät
4. Sämtliche Verbindungskabel

5. MS-DOS Betriebssystem mit speziellem Autoexec für Sofortstart der Textverarbeitung

6. Textverarbeitung »Textmaker« mit Dokumentation bereits voll an die Hardware angepaßt

7. 1000 Blatt Endlospapier Weiß

nur noch zusammenstecken und losschreiben.

Zum Einführungspreis von **DM 1995,-**

Dieses Sonderangebot kann nur im Versandhandel geliefert werden. Lieferung ab Lager, solange der Vorrat reicht. (Falls Lieferung ab Lager nicht möglich ist, erhalten Sie unsere Auftragsbestätigung). Komplette Systeme XT, AT, 386, Portable usw., sowie Einzelkarten, Zubehör und Monitore auf Anfrage. Händler fordern unsere Händlerpreisliste.

ALBATROS-COMPUTER GmbH · 7107 Bad Wimpfen, Magnusstr. 9 · Tel. (0 70 63) 70 57, Fax (0 70 63) 81 91
Ein Name den man sich merken sollte

lich getrennt von dem »normalen« Speicher, wo Programme und Daten abgelegt werden, und diese Trennung spiegelt sich auch in den Adressen wider. Während der Hauptspeicher eines PC von »0000:0000« bis maximal »9000:FFFF« liegt, ist der Bildschirmspeicher der CGA-Karte zwischen Adresse »B800:0000« und »B800:3FFF« angesiedelt und damit 16 KByte groß. Der Bildschirmspeicher der MDA-Karte beginnt dagegen bei »B000:0000« und endet bei »B000:FFFF«, umfaßt also 64 KByte, aufgeteilt in zwei Bildschirmseiten zu je 32 KByte.

Der Basisaufbau des Bildschirmspeichers ist im oberen Teil von Bild 1 zu sehen. Wie abgebildet, besteht jedes angezeigte Zeichen aus zwei Byte. Das erste Byte ist einfach das Zeichen in ASCII, während das zweite Byte das sogenannte Attribut darstellt. Die Zusammenstellung dieses Attributs ist im unteren Teil des Bildes zu finden. Wie man sieht, bestimmt das Attribut mit vier Bit die Farbe des Zeichens und mit drei Bit die Farbe des Hintergrunds. Das letzte Bit bestimmt, ob das Zeichen blinkend dargestellt wird. Das kleine Basic-Programm in Listing 1 führt die verschiedenen Attribut-Bits vor. Starten Sie das Programm und beachten Sie die Ausgabe, um einen schnellen Blick auf die Fähigkeiten der Attribute zu bekommen.

Der prinzipielle Aufbau des Bildschirmspeichers ist in allen Text-Modi gleich. Auf der Adresse »B800:0000« steht das erste ASCII-Zeichen, das in der oberen, linken Ecke des Bildschirms gezeigt wird. In der nächsten Speicherzelle, Adresse »B800:0001«, steht das Attribut-Byte für dieses Zeichen. Eine Zelle weiter, in »B800:0002«, steht schon das zweite ASCII-Zeichen, danach das zugehörnde Attribut, und so geht es weiter.

Die beiden folgenden Formeln ergeben jeweils die Verschiebungen eines Zeichens relativ zum ersten Zeichen:

in Modus 0 oder 1: (Zeile x 80) + (Spalte x 2)

in Modus 2 oder 3: (Zeile x 160) + (Spalte x 2)

Die Adresse des zugehörnden Attribut-Bytes ist immer die Adresse des Zeichens plus 1. In den beiden Bildern 2 und 3 ist der Speicheraufbau in beiden Fällen verdeutlicht.

Der kritische Leser wird jetzt eine Frage stellen wollen. Oben haben wir gesagt, in einem Text-Modus braucht jedes Zeichen 2 Byte im Bildschirmspeicher. Eine schnelle Berechnung zeigt, daß im »hochauflösenden« Modus 2 oder 3 weniger als 4 KByte (25 Zeilen x 80 Zeichen pro Zeile x 2 Byte pro Zeichen ergibt 4000 Byte pro Bildschirm) gebraucht werden,

Modus	Art	Auflösung	Farben	MDA	CGA	MGA	EGA
0	Text	40x25	2 (S/W)	N	J	N	J
1	Text	40x25	16	N	J	N	J
2	Text	80x25	2 (S/W)	J	J	N	J
3	Text	80x25	16	N	J	N	J
4	Grafik	320x200	4	N	J	N	J
5	Grafik	320x200	4 (grau)	N	J	N	J
6	Grafik	640x200	2 (S/W)	N	J	N	J
7	Text	80x25	2 (S/W)	N	N	J	(*)
10	Grafik	640x200	16	N	N	N	J
13	Grafik	320x200	16	N	N	N	J
14	Text	80x25	16	N	N	N	J
15	Grafik	640x350	16	N	N	N	J
16	Grafik	640x350	?	N	N	N	J
MGA	Grafik	720x348	2 (S/W)	N	N	J	(*)

(*) Manche EGA-Karten emulieren diesen Modus

Tabelle 3. Die Betriebsarten der verschiedenen Bildschirm-Adapter

Farbkomponenten					
Intensität	Rot	Grün	Blau	Nr.	Farbe
0	0	0	0	0	Schwarz
0	0	0	1	1	Blau
0	0	1	0	2	Grün
0	0	1	1	3	Türkis
0	1	0	0	4	Rot
0	1	0	1	5	Violett
0	1	1	0	6	Braun
0	1	1	1	7	Hellgrau
1	0	0	0	8	Dunkelgrau
1	0	0	1	9	Hellblau
1	0	1	0	10	Hellgrün
1	0	1	1	11	Helltürkis
1	1	0	0	12	Hellrot
1	1	0	1	13	Lila
1	1	1	0	14	Gelb
1	1	1	1	15	Weiß

Tabelle 4. Die Farben der CGA-Karte

Modus	Turbo Pascal	GW-Basic
0	TextMode (BW40)	SCREEN 0,0 : WIDTH 40
1	TextMode (C40)	SCREEN 0,1 : WIDTH 40
2	TextMode (BW80)	SCREEN 0,0 : WIDTH 80
3	TextMode (C80)	SCREEN 0,1 : WIDTH 80
4	GraphColorMode	SCREEN 1,0
5	GraphMode	SCREEN 1,1
6	HiRes	SCREEN 2

Tabelle 5. Setzen des Modus mit Turbo Pascal und GW-Basic

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 standar Schwarz, kann aber auf jede beliebige Farbe gesetzt werden.

Tabelle 6. Die Farb-Paletten der CGA-Karte

und Modus 0 oder 1 verlangt sogar nur 2 KByte für den ganzen Bildschirminhalt. Jedoch hat die CGA-Karte 16 KByte Speicher. Was ist mit dem Rest? Nun, der restliche Speicher ist nicht verloren. Es werden sogenannte Bildschirmseiten angeboten, die jeweils einen kompletten Bildschirm darstellen. Natürlich ist nur eine Seite auf einmal sichtbar, jedoch mit ein bißchen Mühe kann eine beliebige Seite zu jedem beliebigen

Zeitpunkt geschrieben werden. Da die Modi 0 und 1 etwa 2 KByte pro Bildschirmseite brauchen, werden acht Seiten (mit Nummern von 0 bis 7) eingerichtet, während in den Modi 2 und 3 vier Seiten (Nummern von 0 bis 3) zur Verfügung stehen (Bilder 2 und 3). Das BIOS bietet eine Funktion an, mit der die aktive Seite gesetzt werden kann. »Aktiv« bedeutet hier, daß diese Seite auf dem Bildschirm ausgegeben wird. Die anderen

Seiten können nach wie vor angesprochen werden, sind aber unsichtbar, bis eine neue aktive Seite gesetzt wird. Diese BIOS-Funktion ist auch in GW-Basic, als Teil des Screen-Befehls zu finden. Eine Anweisung wie `screen , , 0, 1`

bewirkt, daß die Seite 1 auf dem Bildschirm gezeigt wird, während alle Print-Befehle ihre Ausgabe auf Seite 0 schreiben.

Leider findet sich in Turbo Pascal 3.0 keine solche nützliche Funktion, und deshalb ist in Listing 2 die Pascal-Funktion »SetScreenPage« aufgelistet. Diese Routine überprüft die angegebene Seitennummer und selektiert, falls die Nummer in Ordnung ist, die neue Seite und gibt den Wert »True« zurück. Falls die Seitennummer nicht in Ordnung ist, gibt die Routine »False« zurück, ohne eine neue Seite zu aktivieren.

Eine Funktion, die sowohl im BIOS als auch in den höheren Programmiersprachen fehlt, ist »aktive Seite abfragen«. Glücklicherweise speichert das BIOS die Nummer der aktuellen Seite, so daß Sie nur von der entsprechenden Adresse ein Byte ablesen müssen, um diese brauchbare Auskunft zu bekommen. Die Adresse ist »0040:0062« hexadezimal und kann sehr einfach abgefragt werden. In Turbo Pascal ist lediglich eine Zeile

```
Seite := Mem [$40:$62];
```

notwendig, in GW-Basic ist die Sache länger, aber genauso einfach:

```
def seg = &H40
seite = peek(&H62)
def seg
```

Übrigens ist der Befehl »def seg« nach dem Peek-Befehl reine Höflichkeit und setzt das Segment für weitere Peek- und Poke-Befehle zurück auf das Standard-Datensegment des Basic-Programms.

Jetzt möchten Sie vielleicht fragen:

»Wenn nur eine Bildschirmseite sichtbar ist, wozu sind die anderen gut?«. Eine Antwort ist, daß ein sehr schneller Bildschirm-Wechsel erzielt werden kann, wenn das neue Bild stets auf einer anderen Seite fertig steht. Man braucht lediglich die neue Seite einzuschalten, und schon ist das Bild da. Das kleine Basic-Programm in Listing 3 zeigt Ihnen, wie schnell ein Seiten-Wechsel erfolgt. Tippen Sie doch das Programm einfach ein und probieren Sie es aus - es wird Sie überzeugen.

Das Schreiben auf dem Bildschirm ist Bestandteil jeder Sprache. Jedoch ist es umständlich von einer Hochsprache aus, wenn man Farbe, Position (Zeile und Spalte) und Seite auf einmal setzen will. Aber auf der »niedrigen« Ebene des BIOS ist dies nicht schwierig.

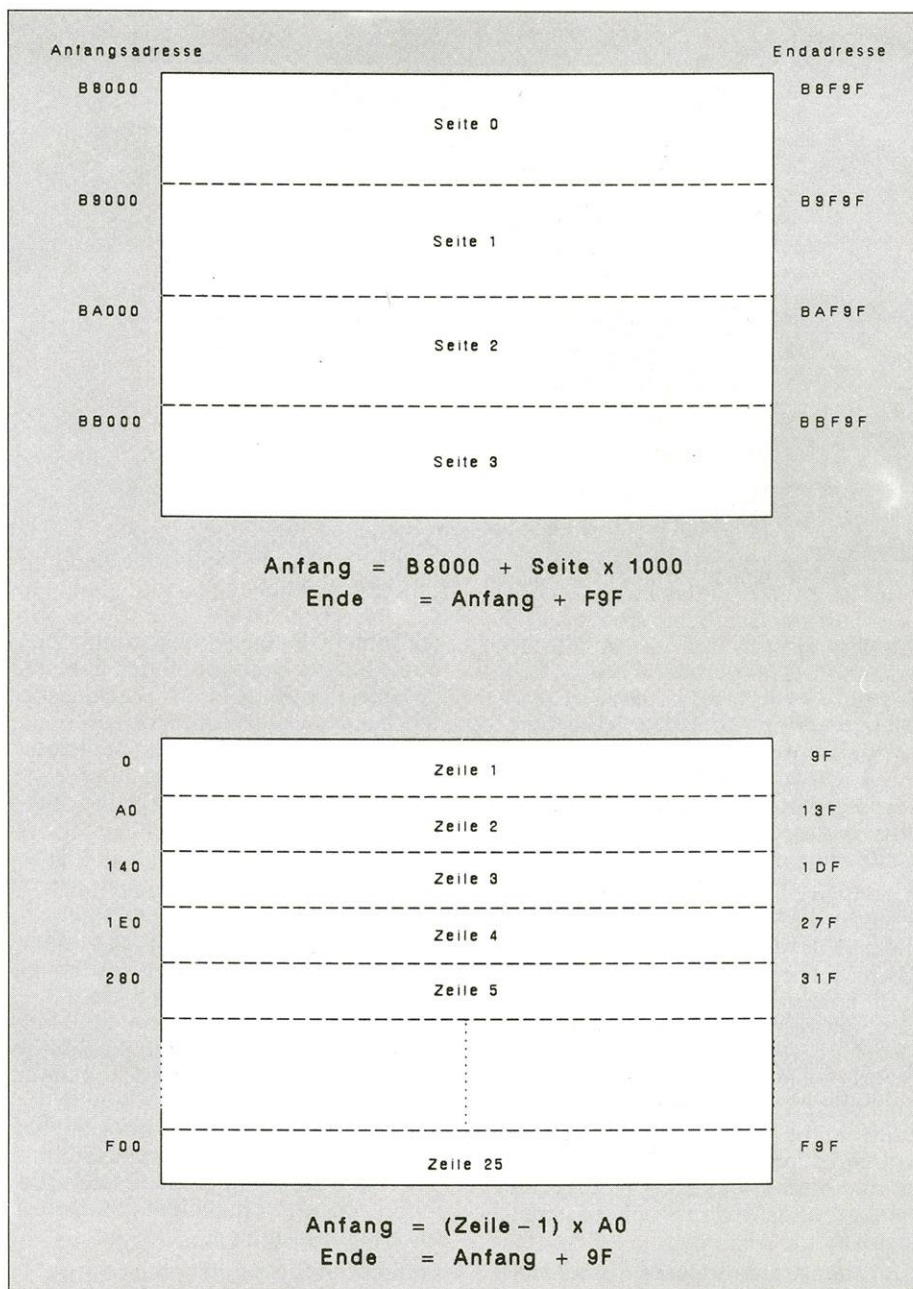


Bild 3. Zeilen und Seiten in den Text-Modi 2 und 3

Um ein Zeichen zu schreiben, bietet das BIOS drei verschiedene Funktionen an. Eine davon, die Nummer 14 »TTY-Schreiben«, ist im wesentlichen die bekannte Write- oder Print-Anweisung. Diese Funktion brauchen wir nicht weiter zu behandeln. Mit den anderen beiden BIOS-Routinen kann man in einem Befehl das Zeichen, dessen Attribut und die Bildschirmseite auf einmal bestimmen. Die Funktion 9 »Zeichen und Attribut schreiben« bekommt vier Parameter: das Zeichen, dessen Attribut, die Seite und die Anzahl zu schreibender Zeichen.

Die Funktion 8 übernimmt das bestehende Attribut und arbeitet nur mit den anderen drei Parametern. In Listing 4 ist eine Turbo-Pascal-Routine »WriteChar« zu sehen, die diese

Funktionalität über das BIOS realisiert. Wenn man aber diese BIOS-Routinen anschaut, wird es klar, daß die beiden Routinen nicht mehr tun, als in den Bildschirmspeicher zu schreiben. Und das können Sie auch gleich direkt tun.

Zunächst die Frage, was passiert, wenn man ein ASCII-Zeichen direkt in den Bildschirmspeicher schreibt? Die erfreuliche Antwort ist diese: Wenn Sie die oben angegebenen Formeln beachten, erscheint das Zeichen auf dem Bildschirm. Das gleiche gilt für Attribute: Wenn Sie ein Attribut-Byte in den Bildschirmspeicher schreiben, bekommt das zugehörige Zeichen sofort die entsprechenden Eigenschaften.

Also gut, doch wie macht man das im einzelnen?

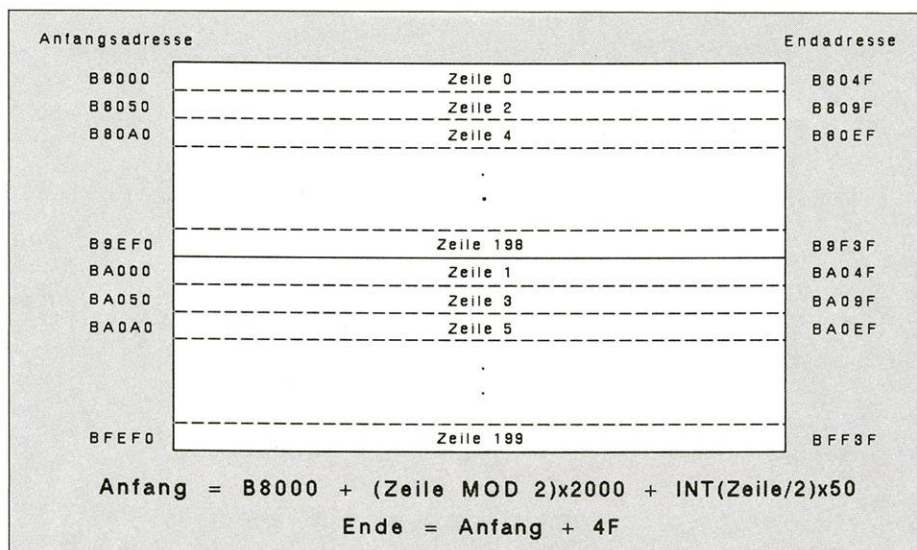


Bild 4. Der Aufbau des Bildschirmspeichers im Grafik-Modus

Erweitern wir zuerst die beiden Formeln, die Speicheradressen aus Spalte und Zeile berechnen, so daß auch die Seite berücksichtigt wird. Dann haben Sie:

In Modus 0 oder 1: (Seite * 2048) + (Zeile * 80) + (Spalte * 2)
 In Modus 2 oder 3: (Seite * 4096) + (Zeile * 160) + (Spalte * 2)

Wie vorher ist die Adresse des Attribut-Byte stets die Adresse des Zeichens plus eins. Jetzt haben Sie alle Voraussetzungen für eine allgemeine Zeichen-Ausgabe-Routine zusammen. Obwohl GW-Basic dies kann, ist unsere Methode einfacher – und Turbo Pascal ist erst gar nicht in der Lage dazu. Deswegen sind die beiden Routinen in Listings 5 und 6 Ihre Aufmerksamkeit wert. Einmal in Basic und einmal in Pascal werden hier Zeichen und Attribute beliebig ausgegeben.

Eine ebenso interessante Funktion ist das Lesen eines Zeichens von einer bestimmten Bildschirmposition. Wie-

der bietet GW-Basic diese nützliche Funktion als Bestandteil des Screen-Befehls, während Turbo Pascal 3.0 wie bei den Bildschirmseiten schweigt. Und wie bei der Ausgabe geht es auch hier einfacher und schneller mit direkten Zugriffen auf den Bildschirmspeicher.

Da die Routinen praktisch identisch mit den Ausgabe-Routinen sind, brauchen sie keine weitere Erklärung. In Listing 7 ist die Basic-Routine, in Listing 8 die Pascal-Routine zu finden. Beide geben das gelesene ASCII-Zeichen in der Variablen »ZCode« und das gelesene Attribut in der Variablen »ZAttr« zurück. Damit haben Sie die wichtigsten Funktionen in den Text-Modi schon gesehen. Alle Funktionen liefern in allen Modi das erwünschte Ergebnis. Passen Sie lediglich auf, daß die Seite, Zeile und Spalte stimmen.

Betrachten Sie zuletzt eine Gruppe von Funktionen, die mit dem Cursor zu tun haben. Die erste setzt die Posi-

tion des Cursors, bewegt also den Cursor zu der gewünschten Stelle, und die zweite fragt die Position des Cursors ab. Beide sind bereits in Hochsprachen zu finden: »GotoXY« in Turbo Pascal und »Locate« in GW-Basic setzen die Cursor-Position, und »WhereX« und »WhereY« (Pascal) beziehungsweise »CsrLin« und »Pos« (Basic) geben die aktuelle Cursor-Position zurück.

Selbstgeschriebene Bildschirmroutinen für Basic und Pascal beschleunigen die Textausgabe und bieten mehr Komfort

Wie üblich weisen die Routinen der Hochsprachen eine Schwäche auf: Sie arbeiten nur auf der aktuellen Seite, das heißt, man muß eine Bildschirmseite sichtbar machen, um den Cursor bewegen zu können. Natürlich geht es aber auch anders, und der andere Weg ist nicht nur mächtiger, sondern auch schneller. Es geht wieder über das BIOS, das freundlicherweise die Cursor-Positionen auf allen Seiten speichert. An der Adresse 450 hexadezimal beginnt ein Array von sechzehn Bytes, in acht Paaren organisiert. Jedes Paar besteht aus einer Spaltenangabe im ersten Byte und einer Zeilenangabe im zweiten Byte, die zusammen die aktuelle Cursor-Position auf der Seite abbilden. Das Lesen dieser Bytes ersetzt völlig die Funktion »CursorPosition abfragen«, und das Schreiben ersetzt mit einer Ausnahme die Funktion »CursorPosition setzen«. Die Ausnahme ist, daß die neu geschriebene Cursor-Position erst dann in Kraft tritt, wenn auf dem Bildschirm geschrieben wird, während die Funktion »Cursor setzen« sofort wirkt. Wenn Sie die Variable »Seite« mit der Nummer der gewünschten Seite belegt haben, wird die Cursor-Position in GW-Basic so gelesen:

```
def seg = &H40
CurZeile = peek (&H50 + Seite*2 + 1)
CurSpalte = peek (&H50 + Seite*2)
def seg
```

In Turbo Pascal funktioniert es folgendermaßen:

```
CurZeile := Mem [$40 : $50 + Seite*2 + 1];
CurSpalte := Mem [$40 : $50 + Seite*2];
```

Die letzte in diesem Zusammenhang vorzustellende Routine bestimmt die Größe des Cursors. Um diese Routine zu verwenden, muß man wissen, wie ein Zeichen auf dem Bildschirm aus Punkten zusammengestellt wird. Die Punkt-Matrix besteht bei der Farbgrafikkarte aus acht Reihen von je acht Punkten. Die Reihen werden von oben (Reihe 0) nach unten (Reihe 7) numeriert. Mit der BIOS-Routine

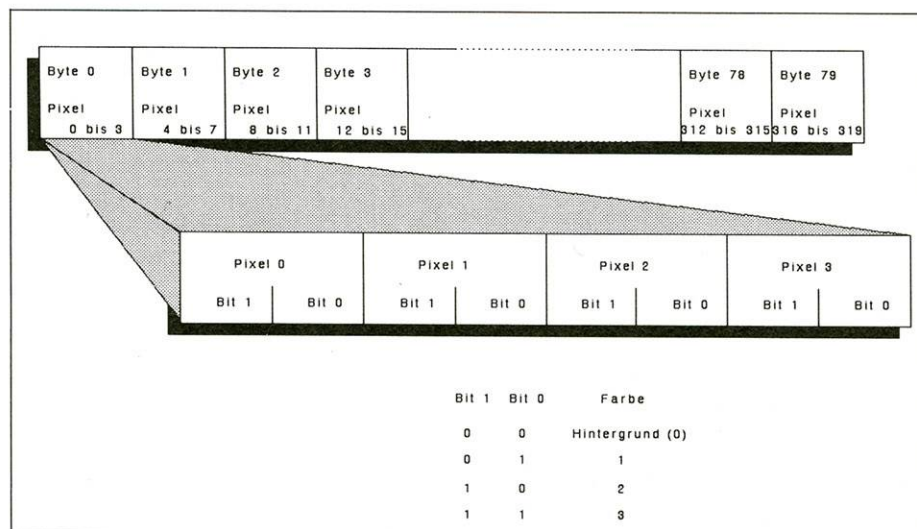


Bild 5. Zeilen und Punkte in den Grafik-Modi 4 und 5

»Cursorgröße setzen« legt man fest, aus welchen Reihen der Cursor besteht. Diese Routine ist im Basic bereits als Teil des Locate-Befehl implementiert. Zum Beispiel erzeugt der Befehl

```
locate,,,7,7
```

den bekannten Unterstrich als Cursor, während der Befehl

```
locate,,,0,7
```

einen Block als Cursor ergibt. Zum Abwechseln könnte man

```
locate,,,3,4
```

ausprobieren. Das Ergebnis ist ein doppelter Strich in der Mitte des Feldes. Auch in Basic kann man problemlos den Cursor ausschalten (unsichtbar machen):

```
locate,,,0
```

schaltet den Cursor ab, und

```
locate,,,1
```

schaltet ihn wieder ein.

Da diese Funktionen in Turbo Pascal fehlen, ist in Listing 9 die Routine »SetCursor« zu finden, die die Größe und Sichtbarkeit des Cursors bestimmt. Ein weiterer Tip: Es gibt keine Funktion, die die aktuelle Größe des Cursors zurückgibt. Hier bietet das BIOS wieder Hilfe. In den zwei Byte auf Adresse 460 hexadezimal stehen die Anfangs- und Endereihen des Cursors. Komischerweise steht das Ende vor dem Anfang, aber wenn man das weiß, ist es kein Problem. Diese beiden Bytes können analog zum Cursor-Position-Array gelesen werden, jedoch hat das Schreiben in diese Bytes keine sofortige Auswirkung.

Die Grafik-Modi der Farbgrafikkarte sind im Detail recht interessant

Wie oben schon erwähnt wurde, ist die Arbeitsweise der CGA-Karte im Grafik-Modus ganz wesentlich anders als in einem Text-Modus. Dieser Unterschied ist auch im logischen Aufbau des Bildschirmspeichers zu sehen (Bild 4). Zunächst sieht der Speicher ganz ordentlich aus, die ersten 80 Byte enthalten die erste Grafikpunktzeile (Zeile Nummer 0). Aber dann muß man die zweite Zeile suchen und suchen und suchen... Die zweite Grafikzeile (Zeile Nummer 1) steht nämlich nicht direkt nach der ersten, wie in den Text-Modi, sondern am Anfang der zweiten Hälfte des Bildschirmspeichers, genauer an der Adresse »B800:2000«. Aber was steht dann nach der ersten Zeile? Na klar, die dritte Zeile! Und nach der zweiten steht die vierte, und nach der dritten die fünfte, und so abwechselnd weiter, bis endlich alle 200 Zeilen im Speicher erfaßt sind. Zusammengefaßt ist der Bildschirmspeicher

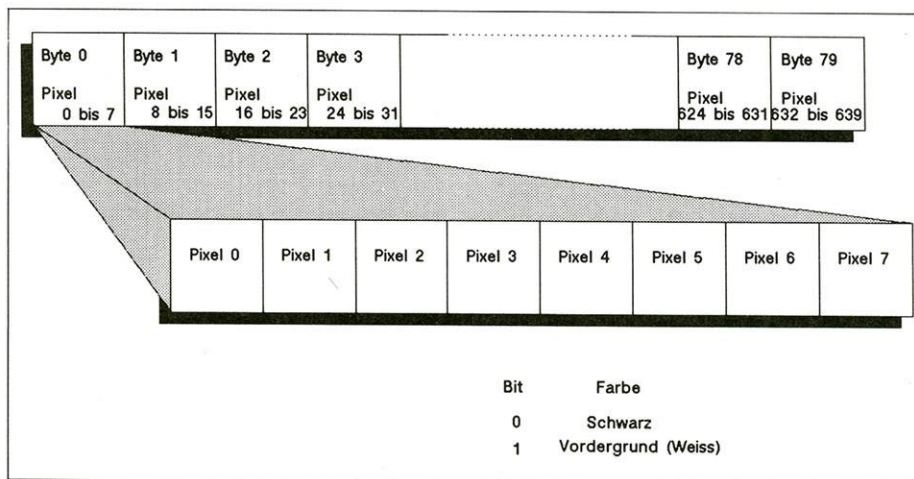


Bild 6. Zeilen und Punkte im Grafik-Modus 6

in den Grafik-Modi in zwei Abschnitte geteilt. Die erste Hälfte beinhaltet alle geraden Zeilen (Nummern 0, 2, 4 etc.), eine nach der anderen, und die zweite Hälfte beinhaltet alle ungeraden Zeilen (1, 3, 5 etc.) in der gleichen Reihenfolge.

Die Frage, wieso der Speicher so (un)organisiert wurde, ist gestattet und bekommt eine allzu übliche Antwort: Die Hardware wurde dadurch ein bißchen einfacher. Daß die Programmierung der Hardware dadurch wesentlich schwieriger wird, und viele Programme dadurch sehr verlangsamt werden, ist dem Hardware-Entwickler natürlich egal. Also müssen die PC-Anwender mit einer verwirrenden Speicherstruktur leben. Für alle Grafik-Modi ist diese Grundstruktur gleich, und zusätzlich ist eine Grafikzeile am Bildschirm immer 80 Byte lang. Wir können daher eine Formel für die Anfangsadresse einer Grafikzeile entwickeln:

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

Da die anderen Eigenschaften des Speichers und die Verwendung der BIOS-Routinen stark vom gewählten Modus abhängen, schauen wir zunächst die Modi 4 und 5 (320 x 200 Pixel in vier Farben beziehungsweise Graustufen) an, und anschließend betrachten wir den hochauflösenden Modus 6.

Der vierfarbige Modus 4 der CGA-Karte ist der beliebteste und am meisten verwendete Grafik-Modus des IBM-PC. Dieser Modus bietet eine relativ hohe Auflösung in mehreren Farben, was kein anderer Modus der CGA-Karte kann, er wird deswegen in allen Bereichen eingesetzt. Wie eine Zeile in diesen Modi gespeichert wird, sehen Sie im Bild 5. Jeder Punkt auf dem Bildschirm wird in zwei Bit abgebildet, deren Wert die Farbe des Punktes bestimmt. Deshalb gibt es in diesem Grafikmodus vier mögliche Farben. Es werden genau

vier Punkte in jedem Byte gespeichert. Die linken vier Punkte einer Zeile im ersten Byte, die nächsten vier im zweiten Byte und so weiter. Bild 5 zeigt, wie die vier Punkte innerhalb eines Bytes abgelegt werden. Damit können wir unsere Formel für die Adresse einer Zeile jetzt erweitern, um die genaue Position eines Punktes zurückzugeben. Hier reicht die Adresse nicht aus, da wir auch wissen müssen, in welchen zwei Bit der Punkt gespeichert wird, das heißt, wir brauchen eine zusätzliche Information. Diese Information berechnen wir als ein Byte, worin genau zwei Bit gesetzt sind und zwar die zwei, die den gesuchten Punkt abbilden. Dieses Byte wird »Maske« genannt. Die Formeln sind:

$$\begin{aligned} \text{Adr} &= (\text{Zeile} \text{ MOD } 2) * 8192 + \text{INT}(\text{Zeile} / 2) * 80 + \text{INT}(\text{Spalte} / 4) \\ \text{Maske} &= 192 / (4 \uparrow (\text{Spalte} \text{ MOD } 4)) \end{aligned}$$

Die letzte Zeile dürfte etwas undurchsichtig sein. Für diejenigen, die sich im hexadezimalen Zahlensystem zu Hause fühlen, sagen wir es so:

$$\text{Maske} = \text{SHR}(\$C0, 2 * (\text{Spalte} \text{ MOD } 4))$$

Beide Formeln für die Maske sind identisch, von der Schreibweise abgesehen. Mit diesen Auskünften können Sie bereits ein paar Programm-Zeilen erfinden, die einen direkten Zugriff auf dem Bildschirmspeicher auch im Grafikmodus implementieren. Es gibt auch BIOS-Routinen, um Pixel zu schreiben und zu lesen, aber da fast alle Sprachen diese Funktionen (»Pset«, »Preset« und »Point« in GW-Basic, »Plot« und »GetDotColor« in Turbo Pascal) schon beinhalten, werden wir nichts mehr darüber sagen. Die entsprechenden Pascal-Zeilen sehen so aus:

```
Adr := (Zeile MOD 2) * 8192 + Int(Zeile/2)
      + INT(Spalte/4);
Farbe := Shl (Farbe, 2*(3-(Spalte MOD 4)));
Maske := Shr ($C0, 2*(Spalte MOD 4));
Mem [$B800:Adr] := Mem [$B800:Adr] AND
(NOT Maske) OR Farbe;
```


Die Berechnung der Adresse und Maske ist schon bekannt, und die Farbe wird analog zur Maske berechnet. Die letzte Zeile macht alles, nachdem die ersten drei Zeilen die Vorbereitungen getroffen haben. In der vierten Zeile wird das Byte aus dem Bildschirmspeicher gelesen, um die anderen drei Punkte wieder schreiben zu können. Mit »Maske« werden die beiden Bits gelöscht, die zu dem zu schreibenden Punkt gehören, und danach werden die Bits wieder entsprechend gesetzt, um die gewünschte Farbe zu bekommen. Das Ganze wird sofort wieder in den Bildschirmspeicher geschrieben, und schon ist alles fertig.

Wenn man einen Punkt nur lesen will, geht man ähnlich vor. Die Adresse und die Maske werden gleich berechnet, die Farbe fällt weg – die wird ja jetzt gesucht! Wenn Sie die Farbe herausgeholt haben, müssen Sie lediglich die beiden Bits ganz nach rechts bringen, um eine Zahl von 0 bis 3 zurückzuliefern. Das Ganze geht so:

```
Adr := (Zeile MOD 2)*8192 + Int(Zeile/2)
+ INT(Spalte/4);
Maske := Shr ($C0, 2*(Spalte MOD 4));
Farbe := Shl (Mem[$B800:Adr] AND (NOT
Maske), 2*(3-(Spalte MOD 4)));
```

Diese beiden Programm-Stücke lassen sich relativ einfach in Basic umsetzen. Nehmen Sie die Routinen aus den Listings 5 und 6 als Vorbild, weil sich in diesen beiden Listings die Sprach-Elemente der beiden Sprachen leicht vergleichen lassen. Eine schnelle Berechnung zeigt, daß in diesen beiden Modi der Speicher praktisch voll ausgenutzt wird. Wir haben $320 \times 200 = 64000$ Punkte, geteilt durch 4 Punkte pro Byte ergibt 16000 Bytes, um ein Bild zu speichern. Von den 16 KByte auf der CGA-Karte sind nur noch 356 Byte übrig – und die werden nicht verwendet.

Jetzt wissen Sie, daß jeder Punkt als zwei Bit gespeichert wird, und wie Sie die Bits im Bildschirmspeicher finden können. Aber wie bestimmen die Bits die Farbe des Punkts, und welche vier Farben von den sechzehn werden verwendet? Diese Fragen werden durch ein neues Konzept, die Farbpalette, beantwortet. Die Palette beinhaltet, wie bei einem Maler, eine Auswahl von allen zur Verfügung stehenden Farben. Bei der originalen IBM-Farbgrafikkarte gibt es zwei Paletten, die Paletten 0 und 1 in Tabelle 6. Inzwischen können fast alle anderen CGA-Karten mit einer von vier Paletten arbeiten, wobei die beiden neuen (Paletten 2 und 3 in der Tabelle) die hellen Versionen der originalen Farbpaletten sind. Ein Basic-Pro-

gramm zur Demonstration der Farbpaletten finden Sie in Listing 10. Wenn vier verschiedene Ergebnisse zu sehen sind, haben Sie eine moderne CGA-Karte. Dies verdeutlicht nebenbei eine interessante Eigenschaft der Paletten: Wenn die Palette gewechselt wird, wechseln sofort alle Farben auf dem Bildschirm.

Eine weitere Eigenschaft aller Paletten stellt die Hintergrundfarbe dar. Diese Farbe hat immer die Nummer 0, aber eine beliebige Farbe kann zur Hintergrundfarbe deklariert werden. Das heißt, in jeder Palette sind drei Farben festgelegt, während die vierte frei wählbar ist. Diese »freie« Farbe ist aber, wie gesagt, die Hintergrundfarbe. Dies bedeutet, daß ein gelöschter Bildschirm stets diese Farbe hat. Die Palette kann man bequem von Pascal oder Basic aus setzen. In Turbo Pascal sind zum Beispiel die Zeilen

```
Graphbackground (White);
Palette (1);
```

alles, was man braucht, um die Farben in Modus 4 zu bestimmen. GW-Basic schafft das Ganze in einer Anweisung:

```
color 15,1
```

Allerdings lehnt Basic es grundsätzlich ab, die Paletten 2 oder 3 zu setzen. Je nach Basic-Version bekommt man eine Fehlermeldung, oder die Anweisung (zum Beispiel »COLOR 15,2«) wird einfach verschluckt. Es gibt einen einfachen Umweg, der unseren einzigen direkten Zugriff auf die Bildschirmsteuerlogik verlangt. Dies geht wie folgt: Zunächst wählen Sie Ihre Hintergrundfarbe aus, danach die gewünschte Palette. Suchen Sie in der folgenden Tabelle die Palette aus, und addieren Sie den nebenstehenden Wert zu der Nummer Ihrer Hintergrundfarbe:

Palette	Wert
0	0
1	32
2	16
3	48

Setzen Sie das Resultat anstelle von »x« in der folgenden Basic-Anweisung ein:

```
out &H3D9,x
```

Und das war es auch schon. Die Farbpalette ist nach Wunsch eingestellt worden. Natürlich gibt es auch eine BIOS-Funktion, um die Palette zu setzen, aber die brauchen Sie jetzt nicht mehr. Wenn man die aktive Palette abfragen will, sieht es finster aus. Weder Turbo Pascal noch GW-Basic bietet solch eine Funktion an, und sogar das BIOS schweigt an dieser Stelle. Nirgendwo in der PC-Literatur wird erwähnt, daß das BIOS oder das MS-DOS die Palette irgend-

wo speichert. Jetzt könnte die logische Folgerung sein: Wenn man oben einen Ausgabe-Befehl verwendet hat, um die Palette zu setzen, könnte man genauso den entsprechenden Eingabe-Befehl verwenden, um die Palette abzulesen. Leider stimmt das nicht. Jeder lesende Zugriff auf die Port-Adresse 3D9hex gibt 255 zurück, egal was vorher geschrieben wurde. Fazit: Sie sollten sich merken, was für eine Palette gerade ausgewählt wurde, da dies nicht abgefragt werden kann. Der letzte Modus der CGA-Karte ist der hochauflösende Modus 6. Hier besteht jede Zeile aus 640 Punkten, das doppelte der Modi 4 und 5. In diesem Modus wird jeder Punkt als ein Bit gespeichert, was die gleiche Ausnutzung des Bildspeichers ergibt, wie bei den anderen beiden Grafik-Modi. Der Aufbau einer Zeile ist ähnlich wie in den Modi 4 und 5 und ist im Bild 6 zu sehen. Die Zeilenadressen sind sogar gleich, da eine Zeile genauso viele Bits braucht wie vorher, jedoch besteht die Maske nur noch aus einem einzigen Bit. Die beiden Formeln für Byte-Adresse und Bit-Maske eines Punktes in Modus 6 sind:

```
Adr = (Zeile MOD 2)*8192 + INT(Zeile/2)*80 + INT(Spalte/8)
Maske = 128 / (2 ↑ (Spalte MOD 8))
```

Wie vorher ist die Formel für die Maske alternativ auch so zu schreiben:

```
Maske = SHR ($80, (Spalte MOD 8))
```

Um Punkte zu lesen oder zu schreiben, können Sie wieder entweder BIOS-Routinen verwenden oder direkte Zugriffe auf den Bildschirmspeicher programmieren. Sowohl GW-Basic als auch Turbo Pascal beherrschen das schon, und zwar mit den gleichen Funktionen wie in den Modi 4 und 5. Wir brauchen uns daher nicht mehr darum zu kümmern. Da es nur ein Bit pro Punkt gibt, gibt es auch nur zwei Farben in diesem Modus. Die Geschichte wird bekannt sein: Die originale IBM-CGA-Karte kannte nur schwarz (Hintergrund) und weiß (Vordergrund) im Modus 6. Inzwischen ist es aber üblich, daß die Vordergrundfarbe frei ausgewählt werden kann. Zu diesem Zweck findet man in Turbo Pascal die Prozedur »HiResColor«, aber GW-Basic und das BIOS bieten hier keine Möglichkeiten. Falls Sie das Glück haben, im Besitz eines neueren PC zu sein, können Sie die folgende Basic-Zeile verwenden, um eine beliebige Vordergrundfarbe im Modus 6 einzusetzen:

```
out &3D9, Farbe
```

Farbe kann jeden Wert von 0 bis 15 annehmen. Eine Umprogrammierung dieses Werts ändert alles auf dem

Bildschirm zur neuen Farbe, was manchmal ein schöner Effekt ist. Die Grafik ist schön und gut, aber meistens will man nicht reine Grafik, sondern eine Mischung aus Grafik und Text. Wie geht das? Zunächst sollten Sie wissen, daß Funktionen wie »print« in Basic oder »WriteLn« in Pascal auch in einem Grafik-Modus funktionieren, das heißt, Sie können von einer Hochsprache aus ohne weiteres Grafik und Text mischen. Die Farbe, in der der Text ausgegeben wird, ist vom Modus und gegebenenfalls von der Palette abhängig. Im Modus 4 ist die Textfarbe stets Farbe 3 der aktuellen Palette, im Modus 5 natürlich die entsprechende Graustufe. Die Textfarbe im Modus 6 ist logischerweise immer die Vordergrundfarbe.

Ein Problem beim Mischen von Text und Grafik ist, daß die Bildschirmkoordinaten für den Text und die Grafik unterschiedlich sind: Für Text verwendet man nach wie vor Reihe und Spalte in einer 40/80 x 25 Textmatrix. Für eine Grafik verwendet man dagegen die X- und Y-Koordinaten des Grafik-Bildschirms, 320/640 x

200. Die Umwandlung ist jedoch nicht sehr schwierig:

Text-Reihe = (Grafik_X/8) + 1
Text-Spalte = (Grafik_Y/8) + 1

Die Formeln gelten für alle drei Grafik-Modi.

Die letzte Funktion in der Bildschirm-Gruppe klingt einfach, aber es steckt viel dahinter. Sie haben schon erfahren, wie man ein Zeichen vom Bildschirm liest, wenn ein Text-Modus aktiv ist. Wie aber kann man dies machen, wenn der aktuelle Modus ein Grafik-Modus ist? Basic macht das Leben auch hier einfach: Die Screen-Funktion gibt das Erwartete zurück, sowohl in einem Text-Modus als auch in einem Grafik-Modus. Hier müssen Sie lediglich wissen, daß die Screen-Funktion auch in einem Grafik-Modus Text-Koordinaten will. Verwenden Sie daher einfach die beiden Formeln oben, um die Grafik-Koordinaten gegebenenfalls umzuwandeln.

Turbo Pascal 3.0 besitzt keine ähnliche Funktion, und die Routine im Listing 8 wird wohl Unsinn zurückliefern, wenn sie in einem Grafik-Modus

aufgerufen wird. Das BIOS beinhaltet aber eine geeignete Routine, und mit der Funktion im Listing 11 steht diese Routine dem Pascal-Anwender zur Verfügung. Das Interessante an dieser Routine wird deutlich, wenn man daran denkt, wie ein Zeichen im Bildschirmspeicher abgelegt wird, wenn ein Grafik-Modus selektiert ist. Das Zeichen wird nämlich, wie alles andere auch, als einzelne Punkte gespeichert. Die Routine muß dann ein Bitmuster aus dem Bildschirm herauslesen, und dieses Muster mit allen verfügbaren Zeichenmustern vergleichen, um das Zeichen zu identifizieren. Dies ist keine leichte Aufgabe. Das BIOS erspart also eine Menge Arbeit an dieser Stelle.

Damit ist unsere Reise durch die Bildschirmsteuerung zu Ende. Was hier geboten wurde, sollte Ihre Arbeit erleichtern, die Grenzen des Möglichen ein bißchen weiter stecken, und – vor allem – viele Anreize für eigene Programme bereitstellen.

(Thomas Little/ev)

```
[TNG] 1 ' ATTR.BAS - Initialisierung, Modus 3, lösche B
ildschirm
[7M4] 10 SCREEN 0,1: WIDTH 80
[CB1] 20 CLS
[FW6] 21 ' Schreibe Text-Muster
[433] 30 FOR I=6 TO 21
[WX4] 40 FOR J=12 TO 72 STEP 4
[C36] 50 LOCATE I,J: PRINT "ABC";
[7D1] 60 NEXT
[2H1] 70 NEXT
[A49] 71 ' Schreibe Attribut-Byte-Werte
[KU4] 80 FOR I=13 TO 73 STEP 4
[B57] 90 LOCATE 4,I: PRINT HEX$((I-13)*4);
[3H1] 100 NEXT
[DC3] 110 FOR I=6 TO 21
[TL6] 120 LOCATE I,3: PRINT HEX$(I-6);
[9H1] 130 NEXT
[9R4] 131 ' Schreibe Titel
[NWC] 140 LOCATE 1,25: PRINT "Attribute in den Text-Mod
i"
[2U8] 141 ' Fange mit Attribut = 0 an
[E72] 150 ATTR = 0
[N7C] 151 ' Setze Bildschirmspeicher als POKE-Ziel
[SD3] 160 DEF SEG = &HB800
[KVG] 161 ' Schreibe die Attribute direkt in Bildschirm
speicher
[TC3] 170 FOR I=6 TO 21
[6L4] 180 FOR J=12 TO 72 STEP 4
[RX5] 190 ADDR = (I-1)*160 + (J-1)*2 + 1
[UPB] 200 POKE ADDR,ATTR: POKE ADDR+2,ATTR: POKE AD
DR+4,ATTR
[KC3] 210 ATTR = ATTR+1
[PS1] 220 NEXT
[AH1] 230 NEXT
[N6C] 231 ' setze SEG zurück (vorsichtshalber...)
[S12] 240 DEF SEG
[6VC] 250 LOCATE 24,25: PRINT "[Ctrl-Break] zu beenden"
;
[X82] 260 GOTO 260
```

Listing 1. Dieses Basic-Programm zeigt alle Attribute

```
[RLF] 0: VAR Regs: RECORD ax,bx,cx,dx,bp,si,di,ds,es
,fl: INTEGER;
[TM0] 1: END;
[000] 2:
[YXC] 3: FUNCTION SetScreenPage (Seite: INTEGER): BO
OLEAN;
[B21] 4: BEGIN
[SN6] 5: SetScreenPage := FALSE;
[HS5] 6: IF (Seite<0) THEN EXIT;
[GP6] 7: IF (Mem[$40:$49]<2) THEN BEGIN
[AT5] 8: IF (Seite>7) THEN EXIT;
[DD0] 9: END
[YY1] 10: ELSE BEGIN
[YS5] 11: IF (Seite>3) THEN EXIT;
[TM0] 12: END;
[6W5] 13: Regs.ax := 1280+Seite;
[XW3] 14: Intr (16,Regs);
[C86] 15: SetScreenPage := TRUE;
[TM0] 16: END;
```

Listing 2. Bildschirmseiten in Turbo Pascal

```
[643] 1 REM PAGE.BAS
[BB1] 10 CLS
[B33] 20 SCREEN ,,1,0
[P17] 40 FOR I=0 TO 1999: PRINT "1";: NEXT
[131] 45 C=1
[933] 50 SCREEN ,,0,0
[XF7] 55 C=((C+1) MOD 8)+1: COLOR C: LOCATE 1,1
[Z73] 60 FOR I=0 TO 1999
[CD7] 70 PRINT "0";: IF INKEY$<>"" THEN 100
[Z21] 80 NEXT
[P22] 90 GOTO 50
[R13] 100 SCREEN ,,1,1
[MT4] 110 IF INKEY$<>"" THEN 50
[382] 120 GOTO 110
```

Listing 3. Schneller Seitenwechsel in GW-Basic


```
[RLF] 0: VAR Regs: RECORD ax,bx,cx,dx,bp,si,di,ds,es
      ,fl: INTEGER;
[TM0] 1: END;
[000] 2:
[5KK] 3: PROCEDURE WriteChar (Zeichen: CHAR; Attribu
      t, Seite, Anzahl: INTEGER);
[UK3] 4: VAR i: INTEGER;
[B21] 5: BEGIN
[G78] 6: Regs.ax := $900 + Ord(Zeichen);
[LFA] 7: Regs.bx := Lo(Seite)*256 + Lo(Attribut);
[OA5] 8: Regs.cx := Anzahl;
[XW3] 9: Intr (16, Regs);
[TM0] 10: END;
```

Listing 4. Zeichen und Attribute in Turbo Pascal ausgeben

```
[5GM] 0: PROCEDURE WriteChar (Zeichen: CHAR; Attribu
      t, Zeile, Spalte, Seite: INTEGER);
[UK3] 1: VAR i: INTEGER;
[B21] 2: BEGIN
[1TF] 3: IF (Zeile<1) OR (Spalte<1) OR (Seite<0) O
      R (Zeile>25) THEN EXIT;
[UL3] 4: i := Mem [$40:$49];
[LT3] 5: IF (i<2) THEN BEGIN
[ED1] 6: i := 1;
[M79] 7: IF (Spalte>40) OR (Seite>7) THEN EXIT;
[DD0] 8: END
[YY1] 9: ELSE BEGIN
[MD1] 10: i := 2;
[W79] 11: IF (Spalte>80) OR (Seite>3) THEN EXIT;
[TM0] 12: END;
[7GA] 13: i := Seite*i*2048 + (Zeile-1)*i*80 + (Spa
      lte-1)*2;
[8F7] 14: Mem [$B800:i] := Ord(Zeichen);
[P08] 15: Mem [$B800:i+1] := Lo(Attribut);
[TM0] 16: END;
```

Listing 5. Direkter Bildschirmspeicher-Zugriff mit Turbo Pascal

```
[GA4] 9999 REM SCHREIB.BAS
[AGB] 10000 IF (Z<1) OR (SP<1) OR (S<0) OR (Z>24) THEN
      RETURN
[7ZC] 10010 DEF SEG = &H40: MM=PEEK(&H49): DEF SEG: IF
      MM>3 THEN RETURN
[LD4] 10020 IF MM>1 THEN 10040
[XUA] 10030 MM:=1: IF (S>7) OR (SP>40) THEN RETURN ELSE
      10050
[AC8] 10040 MM:=2: IF (S>3) OR (SP>80) THEN RETURN
[E24] 10050 DEF SEG = &HB800
[KDC] 10060 MM:=S*MM*2048+(Z-1)*MM*80+(SP-1)*2: POKE MM,
      C: POKE MM+1,A
[ZB2] 10070 DEF SEG
[V82] 10080 RETURN
```

Listing 6. Zeichen und Attribute mit GW-Basic ausgeben

```
[4H3] 9999 REM LIES.BAS
[HS2] 10000 C=-1: A=-1
[EGB] 10010 IF (Z<1) OR (SP<1) OR (S<0) OR (Z>24) THEN
      RETURN
[BZC] 10020 DEF SEG = &H40: MM=PEEK(&H49): DEF SEG: IF
      MM>3 THEN RETURN
[YD4] 10030 IF MM>1 THEN 10050
[6VA] 10040 MM:=1: IF (S>7) OR (SP>40) THEN RETURN ELSE
      10060
[EC8] 10050 MM:=2: IF (S>3) OR (SP>80) THEN RETURN
[K24] 10060 DEF SEG = &HB800
[KFC] 10070 MM:=S*MM*2048+(Z-1)*MM*80+(SP-1)*2: C=PEEK(M
      M): A=PEEK(MM+1)
[3C2] 10080 DEF SEG
[Z82] 10090 RETURN
```

Listing 7. Zeichen und Attribute in GW-Basic lesen

```
[0LG] 0: FUNCTION ReadChar (Zeile, Spalte, Seite: IN
      Teger): INTEGER;
[UK3] 1: VAR i: INTEGER;
[B21] 2: BEGIN
[1TF] 3: IF (Zeile<1) OR (Spalte<1) OR (Seite<0) O
      R (Zeile>25) THEN EXIT;
[UL3] 4: i := Mem [$40:$49];
[LT3] 5: IF (i<2) THEN BEGIN
[ED1] 6: i := 1;
[M79] 7: IF (Spalte>40) OR (Seite>7) THEN EXIT;
[DD0] 8: END
[YY1] 9: ELSE BEGIN
[MD1] 10: i := 2;
[W79] 11: IF (Spalte>80) OR (Seite>3) THEN EXIT;
[TM0] 12: END;
[7GA] 13: i := Seite*i*2048 + (Zeile-1)*i*80 + (Spa
      lte-1)*2;
[G7B] 14: ReadChar := Mem [$B800:i] + Mem [$B800:i+
      1] * 256;
[TM0] 15: END;
```

Listing 8. Direktes Lesen aus dem Bildschirmspeicher mit Turbo Pascal

```
[RLF] 0: VAR Regs: RECORD ax,bx,cx,dx,bp,si,di,ds,es
      ,fl: INTEGER;
[TM0] 1: END;
[000] 2:
[MND] 3: FUNCTION SetCursor (Anfang, Ende: INTEGER):
      BOOLEAN;
[B21] 4: BEGIN
[XB7] 5: IF (Anfang<0) OR (Ende<0) THEN
[BX3] 6: Regs.cx := $2000
[AR0] 7: ELSE
[568] 8: Regs.cx := (Anfang * 256) + Ende;
[UW3] 9: Regs.ax := $100;
[UZ3] 10: Intr (16, Regs);
[TM0] 11: END;
```

Listing 9. Cursor-Größe in Turbo Pascal setzen

```
[K52] 5 DIM PAL(4)
[869] 6 PAL(0)=0: PAL(1)=&H20: PAL(2)=&H10: PAL(3)=&H30
[WG2] 10 SCREEN 1,0
[CL3] 11 CLS: OUT &H3D9,0
[LG4] 20 LINE (0,0)-(59,59),1,BF
[RL5] 30 LINE (60,60)-(119,119),2,BF
[DW5] 40 LINE (120,120)-(179,179),3,BF
[641] 50 P=0
[1C7] 60 LOCATE 1,24: PRINT "Palette";P;
[32A] 70 LOCATE 2,14: INPUT "<RETURN> drucken";IS
[1M8] 80 P=(P+1) MOD 4: OUT &H3D9,PAL(P): GOTO 60
```

Listing 10. Palette 3 und 4 mit GW-Basic

```
[KOF] 0: VAR Regs: RECORD ax,bx,cx,dx,bp,si,di,ds,
      es,fl: INTEGER;
[TM0] 1: END;
[000] 2:
[WD6] 3: FUNCTION ReadChar: CHAR;
[B21] 4: BEGIN
[004] 5: Regs.ax := $0800;
[XW3] 6: Intr (16, Regs);
[678] 7: ReadChar := Chr(Lo(Regs.ax));
[TM0] 8: END;
```

Listing 11. Zeichen vom Bildschirm lesen in Turbo Pascal

DAS GRAFIKWUNDER CGX FÜR DEN PC!

CGX (Color Graphik Extension) ist eine Turbo Pascal-Toolbox der Superlative: Eine Sammlung von 100 leistungsfähigen Prozeduren, mit denen sowohl der Heimprogrammierer als auch der Profi grafisch beeindruckende und benutzerfreundliche Turbo Pascal-Programme schreiben können, ohne auf Spezialkenntnisse oder zeitraubende Assemblerprogrammierung angewiesen zu sein.

Mit CGX können aus ein und demselben Quelltext lauffähige Programme für 3 hochauflösende PC-Grafikmodi erzeugt werden:

- PC 1512 Farbgrafik (640x200 bei 16 Farben)
- EGA Farbgrafik (z.B. PC 1640, Atari) (640x200 bei 16 Farben)
- CGA HiRes-Grafik (640x200 bei 2 Farben)

Damit ist es ohne weiteres möglich, auf dem PC 1512 Software für EGA-Rechner zu entwickeln – und umgekehrt!

Die CGX-Prozeduren arbeiten sehr schnell und effektiv, da sie unter Verwendung modernster Grafikalgorithmik zu einem großen Teil in kompakten 8086 Inline-Assemblercode realisiert wurden. Sie sind in mehreren Bibliotheken zusammengefaßt, die als Include-Files problemlos in eigenen Programmen verwendet werden können:

Die **KERNEL-Bibliothek** stellt den CGX-Betriebssystem-Kern dar und enthält u. a. einen neuen Bildschirmtreiber, der verschiedene Schrifttypen und -größen zur Verfügung stellt, weiterhin Unterstreichen, Fettschrift, Exponenten und Indizes sowie die Textausgabe auf Grafikkoordinaten, Transparent- und XOR-Modus und vieles mehr.

Die **GRAPHIK-Bibliothek** enthält zahlreiche Grafikprozeduren, angefangen bei »Plot« und »Draw« bis hin zu Ellipsenbögen und Rechtecken mit runden Ecken. Weiterhin finden Sie hier eine ganze Palette von Füllprozeduren für Muster und Farben sowie Prozeduren zum Verzerrern bzw. Biegen von Bildschirmbereichen.

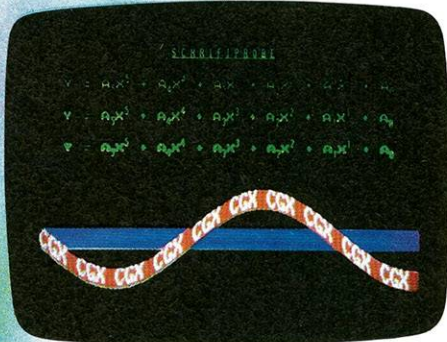
Die **MOUSE-Bibliothek** stellt die Schnittstelle zum Maustreiber dar und erlaubt neben der Abfrage und Kontrolle der Mausposition und -tasten auf komfortable Weise die Gestaltung eigener Mauszeiger. Zusätzlich gibt sie Hilfen zur »Event«-Verarbeitung: »Mauseignisse« können definiert und abgefragt werden.

Die **HARDCOPY-Bibliothek** ermöglicht den Ausdruck beliebiger Bildschirmausschnitte in frei wählbarem Format, wobei die 16 Farben durch verschiedene Muster dargestellt werden.

Die **SCREEN-Bibliothek** enthält Prozeduren, mit denen Bildschirmbereiche im Speicher oder auf Diskette abgelegt und bei Bedarf wieder geladen werden.

Die **WINDOW-Bibliothek** stellt ein System zur Verwaltung von Text- und Grafikenstern zur Verfügung – die Grundlage für eine moderne Menueintechnik.

Weiterhin umfaßt CGX neben Anwendungsbeispielen und einem ausführlichen Handbuch kommentierte Quellprogramme für einen Füllmuster- und Zeichensatzeditor – eine komfortable Programmierhilfe mit »Pull-Down«-Menues und Mausbedienung.



Neue Gestaltungsmöglichkeiten
– auch hier überzeugt CGX

Autor: M. Uphoff

CGX für MS-DOS PC unter Turbo Pascal

Farbenvielfalt durch CGX – mehr als 100 Farben am PC!

199,– DM Best.-Nr. 224

TEXTVERARBEITUNG, TABELLENKALKULATION LOHNBUCHHALTUNG-STATISTIK.....

Überarbeitet? Gönnen Sie sich eine Urlaubsreise in das Land der Fraktale, genießen Sie die phantastische Farbfähigkeit Ihres PC.

– eine neue Dimension für die Freunde fraktaler Grafiken: Über 100 (!) Farben stellt Ihnen dieses Programm bei einer Auflösung von 640 x 200 Punkten zur Verfügung. Und das berühmte »Apfelmännchen« braucht dank ausgefeilter Algorithmen nur noch ca. 3 Minuten für seine Entstehung – keine stundenlangen Wartezeiten hindern Sie mehr daran, eine märchenhafte Welt voller Farben und Formen zu entdecken.

Und dazu der Bedienungskomfort:

- Bedienung per Maus und Pulldown-Menues oder per Cursor
- Hardcopy auf Knopfdruck
- Speichern von Bildern auf Diskette
- Umschalten zwischen verschiedenen Bildern
- Nachträgliches Ändern der Farben
- Vergrößerte Ausschnitte durch Auswahlrahmen
- Rechentiefe bis 9999



FRAKTAL GENERATOR

Autor: Matthias Uphoff

– Bitte benutzen Sie die Bestellkarte –

FRAKTAL GENERATOR

- PC 1512 / PC 1640
- Atari PC
- PC's mit EGA-Karte

FRAKTAL GENERATOR für MS-DOS

- ein CGX-Programm von DMV

nur **49,– DM**

Best.-Nr. 225

**NEU: Jetzt auch für
Cursorsteuerung**



Der PC - glasklar (5)

In diesem Teil geht es um zwei Themen: Zeit und Ton. Wenn man die Zeitverwaltung im PC genau kennt und beherrscht, dann lassen sich die Fähigkeiten des PC um einiges erweitern. Und mit interessanten Geräuscheffekten wird fast jedes Programm lebhafter und hebt sich aus dem Software-Einerlei positiv hervor.

Bisher hat dieser Kurs sich an den verschiedenen Routinen des MS-DOS und des BIOS orientiert, die ein gewisses Standard-Angebot zur Steuerung des PCs darstellen. Diesmal werden Sie ein bißchen tiefer in den PC hineingeführt und sehen, wie man das Betriebssystem und das BIOS erweitern oder gar umgehen kann, um den PC noch effizienter zu verwenden. Es geht dabei in erster Linie um die Timer (Zeitgeber) und die Soundfähigkeiten des PC. Da weder MS-DOS noch das BIOS viel Unterstützung bietet, müssen Sie hier fast alles selber schaffen. Dazu einen kurzen Ausflug in die Mathematik, wo Sie die Beziehungen zwischen Noten, Frequenzen und Zeitgeber-Programmierung kennenlernen. Zu guter Letzt setzen Sie Ihre neuen Kenntnisse in ein Programm um.

Beim letzten Ton des Zeitzeichens war es genau 1441440 Ticks nach Mitternacht...

Bevor Sie sich mit dem Mechanismus der Zeitmessung befassen, müssen Sie etwas mehr über Hardware-Interrupts wissen. In diesem Kurs wurde die Interrupt-Vektortabelle und deren

Nutzung in einem Programm bereits ausführlich erklärt. Um die Zeit zu messen, kommt diese Tabelle wieder ins Spiel, aber jetzt über einen sogenannten Hardware-Interrupt. Wie funktionieren die Hardware-Interrupts? Auf dem Mikroprozessor im PC sind zwei Leitungen für Interrupt-Verarbeitung reserviert. Die erste heißt »Interrupt Request« (deutsch: Forderung auf Unterbrechung) und ist mit allen peripheren Bestandteilen fest verbunden, die einen Interrupt auslösen können. Wenn eines von diesen peripheren Geräten die Aufmerksamkeit des Prozessors braucht (weil es Daten gelesen hat oder ähnliches), aktiviert es die Interrupt-Request-Leitung. Diese Aktion sagt dem Prozessor: »Halt' mal an, jemand braucht dich«.

Da der Prozessor nur eine Leitung für alle solche Anforderungen hat, reagiert er jetzt, indem er die Frage stellt: »Wer braucht mich denn?«. Dies erfolgt dadurch, daß der Prozessor die zweite Interrupt-Leitung, »Interrupt Acknowledge« (deutsch: Bestätigung der Unterbrechung) genannt, aktiviert. Das periphere Gerät versteht diese Leitung als eine Nach-

richt vom Prozessor, »Ich höre, wer bist du?«.

Das periphere Gerät antwortet jetzt mit seiner Interrupt-Nummer. Genau wie bei einem Software-Interrupt nimmt der Prozessor diese Nummer und holt sich die Adresse einer Interrupt-Routine aus der Vektortabelle heraus. Ebenfalls wie beim Software-Interrupt rettet der Prozessor die Flags und die aktuelle Adresse im unterbrochenen Programm auf dem Stack und springt danach zur Interrupt-Routine.

Diese kurze Beschreibung erklärt keineswegs alles über Hardware-Interrupts, jedoch reicht sie für unsere Zwecke aus.

Es wird nun keine Überraschung sein, daß die Zeit im PC über den Hardware-Interrupt-Mechanismus aktualisiert wird. Das periphere Gerät, das für diese Funktion zuständig ist, ist ein Zeitgeber-Baustein von Intel, genannt 8253. Dieser Baustein besitzt drei voneinander unabhängige Zeitgeber, wobei einer davon für die Uhrzeit zuständig ist.

Dieser Zeitgeber wird im Normalfall vom Betriebssystem programmiert und die zugehörige Interrupt-Routine ist ebenfalls ein MS-DOS-Bestandteil. Der Zeitgeber wird so programmiert, daß er 18,2 mal pro Sekunde einen Interrupt auslöst. Dieser Wert 18,2 hat keine magischen Eigenschaften,

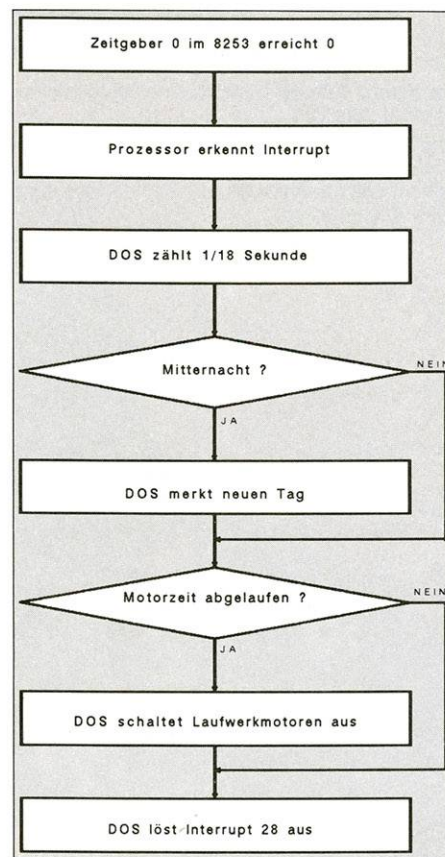


Bild 1. Der Ablauf des Zeitgeber-Interrupts

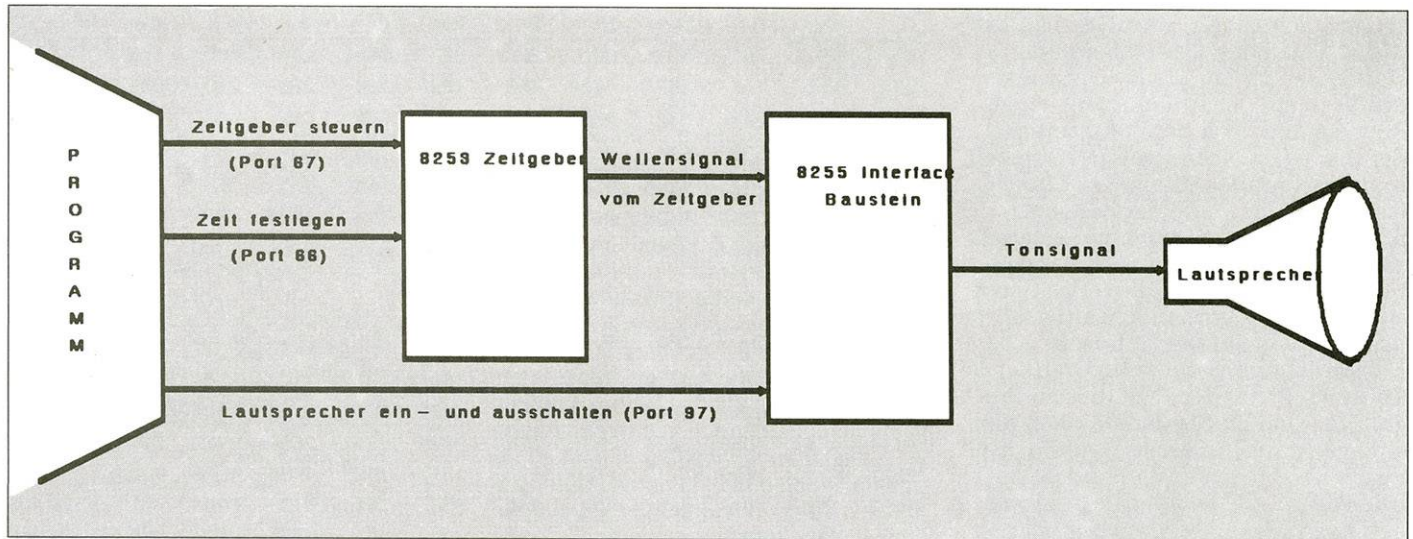


Bild 2. Die Hardware der Tonerzeugung

sondern ist ein Kompromiß zwischen den Fähigkeiten der Hardware und den Wünschen des Programmierers. Wenn dieser Interrupt ausgelöst wird, verzweigt der Prozessor in die Interrupt-Routine, die lediglich einen internen Zähler um eins erhöht.

Dieser Zähler ist schlicht und einfach die MS-DOS-Uhrzeit. Die Zeit wird in der komischen Einheit von 1/18,2 Sekunde, genannt »Tick«, gezählt und wird nur dann in Stunden, Minuten und Sekunden umgewandelt, wenn die Uhrzeit an uns Menschen weitergegeben wird. Als Ausgangspunkt für den Zähler wird Mitternacht verwendet, das heißt, der Wert Null bedeutet null Uhr. Man kann natürlich den Zähler lesen, um die Uhrzeit zu ermitteln und ihn auch schreiben, um die Uhrzeit zu ändern. Der Zähler ist an der Speicheradresse »0000:046C« zu finden und ist vier Bytes lang. Als Beispiel finden Sie eine kleine Turbo-Pascal-Routine im Listing 1, die die Uhrzeit aus diesem Zähler berechnet. Diese Routine entspricht in etwa dem Basic-Befehl »time\$«.

Wenn man schon mal die Uhrzeit hat, möchte man auch vielleicht das Datum bekommen. In GW-Basic ist das keine Kunst, die Funktion »date\$« liefert das gewünschte Ergebnis ohne Programmierung. In Turbo Pascal 3.0 fehlt diese Funktion, jedoch können Sie diesen Mangel mit der Routine in Listing 2 beheben. Sie merken vielleicht, daß diese Pascal-Routine eine MS-DOS-Funktion aufruft. Diese MS-DOS-Funktion ist tatsächlich der einfachste Weg, an das Datum zu kommen, da die Lage des Datums im Speicher sowohl von der MS-DOS-Version als auch von der Größe des Hauptspeichers abhängig ist.

Nun wissen Sie, daß der PC die Zeit »zählt«. Wenn dieses »Zählen« nur

die Uhrzeit und das Datum liefern würden, wäre das Ganze uninteressant. Es gibt aber einiges, das man mit der Zeit machen kann, zum Beispiel warten. Hier müssen Sie unterscheiden zwischen »langen« und »kurzen« Zeiten.

Was alles mit der Zeit geht: Warten, Wiederholen und Messen

In Turbo Pascal kann man mit der eingebauten Standardprozedur »Delay« kurze Zeiten bis etwa 30 Sekunden warten. Die Wartezeit wird in Millisekunden (1/1000 Sekunden) gemessen. In GW-Basic gibt es auch eine (kaum bekannte) Fähigkeit zu warten. Das Kommando »Sound« macht kein Geräusch, wenn die angegebene Frequenz »32767« ist, zum Beispiel

```
sound 32767,91
```

Die zweite Zahl ist die Wartezeit, die in Ticks (etwa 1/18 Sekunde) gemessen wird; die im Beispiel angegebene Zahl »91« entspricht einer Wartezeit von genau fünf Sekunden. Die maximale Wartezeit mit dem Sound-Kommando beträgt ziemlich genau eine Stunde.

Wenn man in Turbo Pascal mehr als 30 Sekunden oder in GW-Basic mehr als eine Stunde warten will, gibt es auch einen direkten Weg, die Wartezeit zu messen. In Basic ist dieser Weg bereits in der Sprache eingebaut, mit der Timer-Funktion, die die Anzahl Sekunden seit Mitternacht liefert. In Listing 3 sehen Sie eine kleine Routine, die eine Wartezeit in Sekunden in der Variable »Warte« akzeptiert und anschließend solange wartet.

Eine äquivalente Routine in Turbo Pascal finden Sie in Listing 4. Weil Turbo Pascal keine Timer-Funktion bietet, wird der MS-DOS-Uhrzeitzähler

direkt gelesen, um diese Funktion zu realisieren.

Natürlich ist Warten nicht das einzige, was mit der Zeit gemacht werden kann. Mit ähnlichen Methoden kann man auch Zeiten messen. Die Timer-Funktion in GW-Basic eignet sich besonders hierfür: Um eine Messung zu beginnen, wird der aktuelle Timer-Wert in einer Variablen gespeichert, dann wird das zu messende Programm ausgeführt und direkt anschließend wird der Timer-Wert wieder gelesen. Der Unterschied zwischen den beiden Timer-Werten ist die vom Programm verbrauchte Zeit in Sekunden. Als Beispiel sehen Sie Listing 5, wo die Zeit für 10000 leere For-Schleifen gemessen wird.

Wie beim Warten geht es in Turbo Pascal sehr ähnlich, nur muß man selber den Zeitzähler lesen und die Ergebnisse in Sekunden umwandeln. Die beiden Routinen »StartTimer« und »ReadTimer« im Beispiel in Listing 6 erfüllen diese beiden Aufgaben; auch dieses Programm mißt die Zeit, die diesmal von 32767 leeren For-Schleifen verbraucht wird.

Eine letzte Anwendung der Zeitfunktionen ist das regelmäßige Wiederholen von Aufrufen. Hier denke man nur an Tastaturabfragen, Tonausgabe oder bewegte Objekte auf dem Bildschirm. Solche Effekte können sehr eindrucksvoll sein, wenn sie anscheinend »nebenbei« passieren, das heißt, wenn das Programm eigentlich etwas ganz anderes tut, aber »gleichzeitig« irgendeinen Spezialeffekt auslöst.

Wiederum ist diese Funktionalität in GW-Basic bereits enthalten. Mit dem Befehl »On Timer« kann man einen regelmäßig wiederholten Aufruf programmieren. Der Befehl sieht so aus:

```
on timer(Zeit) gosub Zeile
```

»Zeit« ist ein Wert zwischen 1 und 86400 und setzt die Wartezeit zwi-

schen sukzessiven Aufrufen auf 1 bis 86400 Sekunden (86400 Sekunden sind genau 24 Stunden). »Zeile« ist die Nummer der Anfangszeile einer Routine, die nach der Wartezeit aufgerufen wird. Nachdem die Wartezeit und die Routine mittels »On Timer« festgelegt sind, muß der Zeitgeber mit dem Befehl »Timer On« gestartet werden; anschließend wird der Aufruf regelmäßig ausgeführt. Das Ganze stellt Listing 7 vor; gleichzeitig soll das Programm einen Anreiz für selbstprogrammierte, tolle Grafik-Routinen darstellen. Mit diesem Programm schaffen Sie den Rahmen für eine vollständige Sprite-Routine. Ein Sprite ist eine Grafikfigur auf dem Bildschirm, die beliebig bewegt werden kann. Das Programm bewegt einen Buchstaben über die Bildfläche. Die Richtung bestimmen Sie mit den Pfeiltasten. Mit der Leertaste können Sie den Buchstaben anhalten, er wird dann blinken (1 mal pro Sekunde, natürlich). Besonders wichtig: Der Bildschirminhalt wird vom Sprite natürlich nicht zerstört, da der Hintergrund gerettet und restauriert wird. So ein Programm in Turbo Pascal zu erstellen, ist wesentlich schwieriger, da Turbo Pascal keine ähnlichen Funktionen besitzt. Natürlich können Sie diese Funktionen selbst »erfinden«, aber um dies zu schaffen, sollten Sie zunächst die MS-DOS-Zeitverwaltung ein bißchen näher betrachten.

In der Zeitverwaltung von MS-DOS ist das Einklinken einer Anwender-Routine vorgeesehen

Die MS-DOS-Zeitroutine macht einiges mehr, als nur den Zähler zu erhöhen. Zum einen ist sie auch dafür zuständig, die Disketten-Motoren auszuschaalten, wenn kein Zugriff innerhalb einer vorprogrammierten Zeit stattfindet. Zum anderen (und jetzt wird es interessant), ruft sie automa-

C	C#	D	D#	E	F	F#	G	G#	A	A#	H
65	69	73	78	82	87	92	98	104	110	116	123
131	139	147	156	165	175	185	196	208	220	233	247
262	278	294	312	330	350	370	392	416	440	466	494
524	556	588	624	660	700	740	784	832	880	932	988
1048	1112	1176	1248	1320	1400	1480	1568	1664	1760	1864	1976
2096	2224	2352	2496	2640	2800	2960	3136	3328	3520	3728	3952
4192	4448	4704	4992	5280	5600	5920	6272	6656	7040	7456	7904

Tabelle 1. Frequenzen und musikalische Notenwerte

C	C#	D	D#	E	F	F#	G	G#	A	A#	H
18357	17292	16345	15297	14551	13715	12969	12175	11473	10847	10286	9701
9108	8584	8117	7649	7231	6818	6450	6088	5736	5424	5121	4831
4554	4292	4058	3824	3616	3409	3225	3044	2868	2712	2560	2415
2277	2146	2029	1912	1808	1705	1612	1522	1434	1356	1280	1208
1139	1073	1015	956	904	852	806	761	717	678	640	604
569	537	507	478	452	426	403	380	359	339	320	302
285	268	254	239	226	213	202	190	179	169	160	151

Tabelle 2. Zählerwerte für den Zeitgeberbaustein und musikalische Notenwerte

tisch eine Anwender-Routine auf. Die Adresse dieser Routine wird in den Interrupt-Vektor 1Chex (dezimal 28) eingetragen, der für diesen Zweck vorgesehen ist. Normalerweise zeigt dieser Vektor auf einen Iret-Befehl (»Rückkehr vom Interrupt«) und bewirkt damit nichts. Das können Sie aber ändern und so wird die Funktionalität von »On Timer« in Turbo Pascal realisiert.

Wenn Sie an diese Stelle eine eigene Routine einhängen, muß die Routine drei Regeln beachten. Erstens, beim Eintritt in die Routine zeigt das Register DS auf die BIOS-Daten im Speicher, einen anderen gewünschten Wert muß die Routine selbst laden; zweitens, alle Register außer AX, DX, DS und die Flags müssen beim Verlassen der Routine unverändert sein; und drittens, die Routine darf keine BIOS-Funktion aufrufen. Das mag sehr einschränkend klingen, aber in der Praxis sind diese Regeln kaum zu spüren.

Bevor nun eine Beispiel-Routine vorgestellt wird, sollte zunächst der

grundsätzliche Ablauf eines Zeitgeber-Interrupts verdeutlicht werden. Bild 1 zeigt den Ablauf des Interrupts vom Zeitgeber-Baustein bis zum Ende der Interrupt-Routine, wobei Sie sehen, daß die Anwender-Routine ganz am Ende der DOS-Routine aufgerufen wird. Dies bedeutet, daß die Uhrzeit bereits aktualisiert wurde, wenn die Anwender-Routine aufgerufen wird.

Jetzt ein paar Worte, um zu erklären, wie all das passiert. Listing 8 zeigt ein kleines Demo-Programm in Turbo Pascal 3.0. Die Interrupt-Routine in diesem Programm heißt »ShowTime«, da sie nämlich die Uhrzeit auf dem Bildschirm zeigt. BIOS-Aufrufe sind hier verboten, deswegen müssen wir direkt in den Bildschirmspeicher schreiben. Dies hat den zusätzlichen Vorteil, einfacher und schneller zu sein, als irgendeine Routine aufzurufen. Die Inline-Zeilen am Anfang und am Ende der Routine kommen aus dem Turbo-Pascal-Manual (Seite 214 für Version 3.0), und sind ein wenig geändert worden, um sie an die Regeln für eine Zeitgeber-Routine anzupassen. Die beiden Zeilen sorgen dafür, daß die Register durch unsere Routine nicht verändert werden, und daß sich die Routine ordentlich beendet.

Das Hauptprogramm hat nur zwei Aufgaben, erstens die Adresse der Interrupt-Routine in die Vektortabelle einzutragen und zweitens, die Routine im Speicher festzumachen und die Kontrolle zurück an das MS-DOS zu geben. Die Adresse der Routine wird in die Tabelle unter Interrupt-Sperre eingetragen. Der Grund dafür ist, daß die Manipulation dieses Vektors in zwei Schritten gemacht werden muß. Es kann dann passieren, daß ein Timer-Interrupt gerade in dem Moment

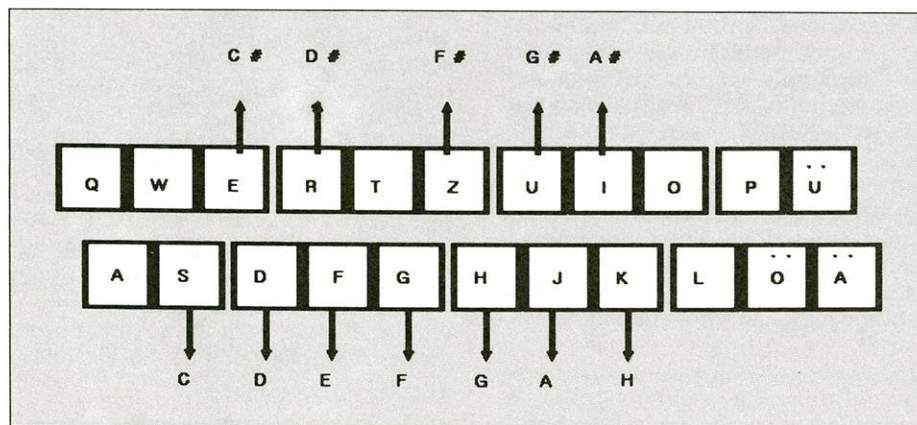


Bild 3. Tastaturbelegung des Klavier-Programms

kommt, wenn wir den Vektor ändern. Falls Sie nur die Hälfte des Vektors geschrieben haben, zeigt der gesamte Vektor irgendwohin, wo es kein sinnvolles Programm gibt. Der Interrupt in diesem Moment führt dann zu völligem Chaos, mit hoher Wahrscheinlichkeit sogar zu einem kompletten System-Absturz. Mit der Interrupt-Sperre schließen wir diesen unangenehmen Fall einfach aus.

Das Beispiel im Listing 8 funktioniert wie das Basic-Programm aus Listing 7, allerdings wird die Routine »MoveSprite« über die MS-DOS-Zeitverwaltung statt mit dem »On Timer« Befehl aufgerufen. Die schnellere Zeitverwaltung von MS-DOS ermöglicht es, das Sprite wesentlich schneller als mit dem Basic-Programm zu bewegen. Mit den Methoden aus dem vorhergehenden Teil dieses Kurses kann man dieses Programm so ändern, daß es in einem Grafik-Modus statt wie hier in einem Text-Modus arbeitet. Damit würden dann echte Grafik-Sprites realisiert.

Kein Bild – kein Ton: MS-DOS macht das schon...

Natürlich ist der PC nicht als Audio-Gerät der ersten Klasse entwickelt worden. Insbesondere entspricht der Lautsprecher nicht gerade der HiFi-Norm, was sowohl die Qualität als auch die Lautstärke der erzeugten Töne beschränkt. Sie sollten also nicht erwarten, daß der PC so gut wie ein CD-Spieler klingt, aber immerhin kann er ohne großen Aufwand wirklich musikalisch werden.

Vielleicht scheinen auf den ersten Blick die zwei Themen Zeit und Ton weit voneinander entfernt. Jedoch sind sie im PC sehr eng verwandt, weil die Hardware-Komponenten in beiden Bereichen dieselben sind. Im Bild 2 ist die PC-Schaltung zur Lautsprechersteuerung zu sehen. Vom Bild her wird es sofort klar, daß zwei Bausteine für die Töne des PC zuständig sind. Den ersten kennen Sie schon, es ist der Zeitgeber-Baustein 8253. Der zweite ist der Schnittstellen-Baustein 8255, auch von Intel. Wie das Bild zeigt, wird der Lautsprecher über den Schnittstellen-Baustein ein- und ausgeschaltet. Außerdem kann dieser Baustein ein Signal vom Zeitgeber-Baustein zum Lautsprecher weiterschalten. Aber wie werden diese Signale in einen Ton umgewandelt?

Alle Bausteine im PC arbeiten mit zwei Spannungen. Wenn diese Spannungen abwechselnd an den Lautsprecher gesendet werden, wird die Lautsprecher-Membrane erst zusammengezogen (von der höheren Spannung) und danach wieder entspannt (von der niedrigeren Spannung).

Wiederholte Ent- und Anspannung der Membrane bringt den Lautsprecher zum Schwingen, und das Ergebnis ist eine Vibration in der Luft, die wir als einen Ton empfinden. Je schneller diese Vibration, desto höher wird der Ton empfunden.

Es gibt zwei Methoden, womit die Signale an den Lautsprecher geführt werden können. Erstens kann der Schnittstellen-Baustein selbständig Töne erzeugen. Dies ist etwas aufwendig, wie Sie in Kürze sehen werden. Zweitens kann der Zeitgeber-Baustein, den Sie schon kennen, ein Signal über den Schnittstellen-Baustein zum Lautsprecher schicken, und dadurch auch Töne generieren. Dies ist der einfachste und flexibelste Weg, Töne zu generieren, der auch zum Beispiel im Basic-Befehl »play« verwendet wird. Nun wollen wir die beiden Methoden ein bißchen näher betrachten, um deren Vor- und Nachteile kennenzulernen.

Der kürzeste Weg zur Tongenerierung geht vom Programm aus über den Schnittstellen-Baustein 8255, der eine direkte Verbindung zum Lautsprecher hat. Mit dem Befehl

```
Port[97] := Port[97] OR 2;
```

schaltet man beispielsweise in Turbo Pascal den Lautsprecher ein, und mit dem Gegenstück

```
Port[97] := Port[97] AND $FD;
```

wird der Lautsprecher wieder ausgeschaltet. Wenn die beiden Befehle nacheinander wiederholt werden (Listing 9), entsteht ein Ton, dessen Frequenz von der Geschwindigkeit des Programms abhängig ist. Diese einfache Methode hat drei Nachteile. Erstens ist der höchst erzeugbare Ton durch die Programmiersprache begrenzt. Eine langsame Sprache wie Basic kann nur tiefe Töne generieren, und selbst schnellere Sprachen können keine sehr hohen Töne erzeugen. Zweitens ist es sehr schwierig, einen bestimmten Ton zu erzeugen, da die Töne von der Geschwindigkeit einer Programmschleife abhängig sind. Um einen tieferen Ton zu generieren, muß man irgendwie die Zeiten zwischen Ein- und Ausschalten einstellen, und dabei die Laufzeiten im Programm einrechnen. Diese Laufzeiten sind vom jeweiligen Prozessor und von der Taktfrequenz abhängig, so daß die Töne von einem IBM-PC anders sind, als die von einem Schneider-PC, der PC/AT erzeugt wieder andere Töne und so weiter. Drittens ist das Programm voll damit beschäftigt, den Ton zu erzeugen. Das heißt, man kann nicht einmal die Tastatur abfragen, ohne den Ton zu beeinflussen. Zusammen sind diese drei Nachteile Grund genug, auf diese Methode der Tonerzeugung zu verzichten. Jedoch gibt es viele sehr gute Programme,

die den Ton eben auf diese Art generieren. Warum? Die Antwort liegt in der Klangkomplexität, die mit dieser Methode erreicht werden kann. Wie gerade gezeigt wurde, muß man sich um alle Details der Tonerzeugung kümmern, wenn man diese Methode einsetzt. Das heißt aber, daß alle Details der Tonerzeugung gesteuert werden können, man hat die komplette Kontrolle über den erzeugten Ton. Nur so kann man sehr komplizierte Töne (eventuell mit mehreren Stimmen) aus dem PC herauslocken.

Die andere Methode der Tonerzeugung weist keine der obengenannten Nachteile auf. Hier wird der Zeitgeber-Baustein 8253 eingesetzt, um ein Signal an den Lautsprecher zu schicken. Der Baustein muß einmal programmiert werden, um den gewünschten Ton zu generieren. Der Ton wird dann ausgegeben, ohne weitere Eingriffe vom Programmierer zu benötigen, das heißt, das Programm kann andere Dinge tun, zum Beispiel auf dem Bildschirm zeichnen oder die Tastatur abfragen, während der Ton gespielt wird. Weiter ist die Frequenz des Tons lediglich vom Zeitgeber abhängig, so daß auf allen kompatiblen Computern die gleichen Töne generiert werden. Wie Sie oben gesehen haben, ist der Baustein in der Lage, sehr kurze Zeiten sehr genau zu messen. Dies bedeutet, daß Sie mit dieser Methode einen beliebigen Ton erzeugen können, ohne Einschränkung durch die Geschwindigkeit der Programmiersprache.

Um den Zeitgeber-Baustein zu programmieren, gibt man zuerst ein Kommando aus, das dem Baustein sagt, daß er seinen Zähler mit dem dann folgenden Wert laden soll. Dieser Zähler wird dann bei jedem Takt um eins erniedrigt. Wenn der Zähler Null erreicht, wird ein Impuls an den Lautsprecher ausgegeben, während der Baustein den Zähler wieder mit dem programmierten Wert lädt und nochmal zu zählen anfängt. In dieser Weise wird eine Kette von Pulsen generiert, deren Frequenz durch den programmierten Wert festgelegt wird. Es bleibt dann die Frage, welche Frequenz von welchem Wert generiert wird.

Wie oben bereits erklärt wurde, empfindet man die Pulschette als eine einzige Note einer bestimmten Frequenz, wenn der Lautsprecher regelmäßig zusammengezogen und entspannt wird. Eine Frequenz ist lediglich eine Kette von Pulsen (oder in diesem Fall, von An- und Entspannungen) mit einer bestimmten Zeit zwischen sukzessiven Anspannungen. Diese Zeit wird die Periode ge-

nannt. Von der Physik her kennen Sie die Formeln

Frequenz = 1 / Periode
und

Periode = 1 / Frequenz

In Tabelle 1 sind die Frequenzen für die Noten sieben musikalischer Oktaven aufgeführt. Jetzt ist die Frage, wie der Zeitgeber-Baustein programmiert wird, um in den gewünschten Abständen einen Impuls an den Lautsprecher zu schicken.

Der Baustein ist wirklich nur ein Zähler, der eine genaue Eingabe-Frequenz zählt. Er wird mit einem Wert programmiert und zählt dann die Eingabe-Frequenz, bis er die programmierte Anzahl Schwingungen gezählt hat. Wenn diese Zahl erreicht wird, gibt er einen Puls (zum Lautsprecher) aus, und fängt wieder von vorne an. Die Zeit zwischen zwei Ausgabe-Impulsen erfüllt also die Formel

Zeit = Zählerstand / Eingabefrequenz

Da diese Zeit wiederum die Periode unserer Ausgabefrequenz ist, ist die Beziehung zwischen Zähler und Ausgabefrequenz laut den beiden Formeln oben folgendes:

Frequenz = Eingabefrequenz / Zählerstand

Der Zeitgeber wird hardwaremäßig mit der Frequenz 1193180 Hertz getaktet, der Wert eins erzeugt dann ein Signal mit dieser Frequenz. Dies ergibt die Formel

Frequenz = 1193180 / Zählerstand

für die Beziehung zwischen programmiertem Wert (Zähler) und erzeugter Frequenz. In Tabelle 2 finden Sie die entsprechenden Werte für den Zeitgeber, wenn er wie üblich mit 1,19 Megahertz getaktet wird. Je höher der Wert, desto länger muß der Zeitgeber-Baustein zählen, um Null zu erreichen, und desto niedriger wird der Ton. Die Programmierung erfolgt über drei Ausgabe-Kommandos:

```
Port [67] := $B6;
Port [66] := Lo (Note);
Port [66] := Hi (Note);
```

Die byteweise Programmierung ist erforderlich, da der Baustein nur über acht Datenleitungen verfügt, das heißt, er kann nur ein Byte auf einmal lesen.

Wie im Bild 2 zu sehen ist, geht das Signal vom Zeitgeber nicht direkt an den Lautsprecher, sondern zum Schnittstellen-Baustein. Dieser ist dafür zuständig, das Signal (und damit den Ton) vom Zeitgeber weiterzuschalten. Das Weiterschalten des Signals erfolgt mit den Kommandos:

```
i := Port [97];
i := i OR 3;
Port [97] := i;
```

Das Ausschalten des Signals wird mit den drei Befehlen

```
i := Port [97];
i := i AND $FC;
Port [97] := i;
```

realisiert. In beiden Fällen lesen wir den alten Zustand des Ports und ändern nur die beiden Bits, die den Lautsprecher steuern. Die anderen sechs Bit haben nämlich ganz andere Funktionen (zum Beispiel Steuerung der Tastatur), die wir in keinem Fall stören sollten. Diese Warnung gilt übrigens auch für den Zeitgeber-Baustein. Der ist für verschiedene Funktionen im PC zuständig, unter anderem das »Auffrischen« der Daten im Speicher. Wenn man einen Fehler in der Programmierung dieses Bausteins macht, besteht durchaus die Gefahr eines kompletten System-Absturzes. Alle diese Kenntnisse über die Tonerzeugung im PC sind in dem Programm im Listing 10 benutzt, um den PC in ein Klavier umzuwandeln. Den Tasten der mittleren und oberen Reihen sind nach dem Vorbild eines echten Klaviers die Noten zugeordnet (Bild 3). Mit der linken Shift-Taste wird eine Oktave tiefer geschaltet, mit der rechten Shift-Taste eine Oktave höher. Die Return-Taste beendet das Programm.

Das Klavier-Programm sollte wie ein Musikinstrument funktionieren. Ein wichtiger Teil dieser Funktionalität ist, daß eine Note solange gespielt wird, wie man die entsprechende Taste gedrückt hält. Das heißt, der Ton fängt an, wenn die Taste gedrückt wird und hört erst dann auf, wenn die Taste losgelassen wird. Die Tastatur am PC sendet beide Auskünfte an den PC beziehungsweise an die Tastatur-Interrupt-Routine, aber die Betriebssystem-Routine wandelt die Nachricht »Taste gedrückt« in ein Zeichen um und verschluckt die Nachricht »Taste losgelassen«, ohne den Anwender zu benachrichtigen. Es besteht damit keine Auskunft darüber, wie lange die Taste gedrückt wurde, oder ob sie noch gedrückt ist. Um dieses Umwandeln zu umgehen, müssen wir die MS-DOS-Routine durch eine andere ersetzen. Diese neue Routine muß die beiden Auskünfte »Taste gedrückt« und »Taste losgelassen« an das Hauptprogramm weitergeben.

Genau das macht die Interrupt-Routine »GetKey« in Listing 10. Die zweite Zeile der Routine

```
c := Port[KbData];
```

liest eine Information direkt von der Tastatur und schreibt sie in die Pascal-Variable »c«, wodurch das Hauptprogramm diese Information auswerten kann. Dies ist der eigentliche Zweck der Routine, alle anderen Zeilen dienen lediglich der sauberen Interrupt-Verarbeitung. Insbesondere sind die zwei Inline-Zeilen am An-

fang und am Ende der Routine hier wichtig. Eine Interrupt-Routine muß sehr aufpassen, daß die Umgebung des unterbrochenen Programms nicht zerstört wird. Dies bedeutet zum einen, daß die Interrupt-Routine keine Variablen ändern darf, die für das andere Programm reserviert sind und zum anderen, daß die Interrupt-Routine keine Register ändern darf. Das letzte ist fast unmöglich, da ein Programm kaum ohne Register arbeiten kann. Deswegen werden die Register am Anfang der Routine gerettet und am Ende der Routine restauriert. Genau dies passiert in den beiden Inline-Zeilen. Die drei Zeilen in der Mitte der Routine:

```
p := Port[KbCtrl];
Port[KbCtrl] := p OR $80;
Port[KbCtrl] := p AND $7F;
```

stellen eine Besonderheit der Tastaturbehandlung dar. Die Tastatur am PC ist nämlich selber »intelligent« und verlangt vom PC eine Bestätigung, daß eine Information gelesen wurde, bevor sie eine weitere Information schickt. Normalerweise erledigt die MS-DOS-Interrupt-Routine diese Aufgabe, aber hier haben wir die MS-DOS-Routine komplett ersetzt. Deshalb müssen Sie der Tastatur sagen: »Information gelesen, alles klar zum Weitermachen«. Das ist es genau, was diese drei Zeilen tun. Wann immer Sie eine Routine erfinden, die die Tastaturbehandlung des Betriebssystems ersetzt, sollten Sie diese drei Zeilen übernehmen.

Der periphere Interrupt-Kontroller, worüber der Tastatur-Interrupt zum Prozessor kommt, besitzt auch ein bißchen »Intelligenz« und braucht ebenfalls eine Bestätigung, daß alles verarbeitet wurde, bevor er bereit ist, einen neuen Interrupt weiterzugeben. Wie bei der Tastatur, wird dies auch hier normalerweise von der MS-DOS-Routine erledigt, aber in unserem Programm haben wir das Ganze übernommen. Sie müssen dann auch dem PIC das »alles klar« sagen, und dies machen wir mit der Zuweisung

```
Port[IntCtrl] := $20;
```

Diese Zuweisung ist übrigens nicht nur für die Tastatur erforderlich. Wenn Sie irgendeine MS-DOS-Interrupt-Routine ersetzen, deren Interrupt über den PIC zum Prozessor kommt, müssen Sie diese Zuweisung in die Routine eintragen. Wenn diese Zuweisung fehlt, wird nur der erste Interrupt erkannt und danach ist Schluß! Damit sind wir am Ende dieses Kapitels und auch dieses Kurses. Die Einführung in die Zeitverwaltung und Tonerzeugung des PC ist wie der gesamte Kurs wieder einmal die Basis, auf der Sie Ihre eigenen Programme aufbauen können.

(Thomas Little/ev)


```

[OB3] 0: PROGRAM Time;
[000] 1:
[H25] 2: TYPE TimeStr = STRING[8];
[000] 3:
[BS8] 4: FUNCTION Strg(i: INTEGER): TIMESTR;
[B21] 5: BEGIN
[EK9] 6:   Strg := Chr ((i DIV 10)+48) + Chr ((i MOD
      10)+48);
[TM0] 7:   END;
[000] 8:
[046] 9: FUNCTION Time: TIMESTR;
[EE0] 10: VAR
[ROC] 11:   DOSZeit : ARRAY [0..1] OF INTEGER ABSOLU
      TE $40:$6C;
[YC3] 12:   Zeit : REAL;
[6X2] 13:   Stunden,
[122] 14:   Minuten,
[645] 15:   Sekunden : INTEGER;
[B21] 16: BEGIN
[LY9] 17:   Zeit := DOSZeit[0] + DOSZeit[1]*65536
      .0;
[535] 18:   Zeit := Zeit / 18.2;
[119] 19:   Stunden := Trunc (Zeit / 3600);
[578] 20:   Zeit := Zeit - (Stunden*3600.0);
[DU8] 21:   Minuten := Trunc (Zeit / 60);
[AKB] 22:   Sekunden := Trunc (Zeit - (Minuten*60));
[KVG] 23:   Time := Strg(Stunden)+' '+Strg(Minute
      n)+' '+Strg(Sekunden);
[TM0] 24:   END;
[000] 25:
[B21] 26: BEGIN
[XG1] 27:   REPEAT
[MU4] 28:     WriteLn (Time);
[CU4] 29:   UNTIL KeyPressed;
[5L0] 30:   END.

```

Listing 1. Diese Routine realisiert die Time-Funktion in Turbo Pascal

```

[H23] 0: PROGRAM Datum;
[000] 1:
[EU6] 2: TYPE ShortStr = STRING[31];
[000] 3:
[CWA] 4: CONST WochenTag: ARRAY[0..6] OF STRING[3]
[LOB] 5:   = ('Son', 'Mon', 'Die', 'Mit',
      'Don', 'Fre', 'Sam');
[000] 6:
[HRE] 7: VAR Regs: RECORD ax,bx,cx,dx,bp,si,di,ds
      es,fi: INTEGER;
[TM0] 8:   END;
[000] 9:
[000] 10:
[5G7] 11: { Datum vom DOS holen }
[5A4] 12: PROCEDURE Date;
[B21] 13: BEGIN
[304] 14:   WITH Regs DO BEGIN
[3X1] 15:     ax := $2A00;
[913] 16:     MsDos (Regs);
[TM0] 17:   END;
[TM0] 18:   END;
[000] 19:
[000] 20:
[B21] 21: BEGIN
[X81] 22:   Date;
[304] 23:   WITH Regs DO BEGIN
[ZD3] 24:     WriteLn ('');
[PT7] 25:     Write (WochenTag[Lo(ax)], ' ');
[SV9] 26:     WriteLn (Lo(dx):2, '.', Hi(dx):2, '.', cx:4
      );
[TM0] 27:   END;
[5L0] 28:   END.

```

Listing 2. Mit dieser Routine kann der Turbo-Pascal-Programmierer das Datum lesen.

```

[Y2K] 1 'TIMER - Ein kleines Beispielprogramm für Wart
      en in Basic
[Z75] 10 INPUT "Zeit";WARTE
[S4A] 20 PRINT "Ich warte";WARTE;" Sekunden ..."
[IN2] 30 GOSUB 1000
[BF7] 40 PRINT "Ich bin wieder da!"
[791] 50 END
[PM0] 998 '
[D6L] 999 'Mit der eingebauten Funktion TIMER ist das W
      arten einfach
[RW3] 1000 T = TIMER+WARTE
[445] 1010 IF TIMER<T THEN 1010
[T12] 1020 RETURN

```

Listing 3. Warten läßt sich in GW-Basic mit der Funktion »timer« leicht realisieren.

```

[1N4] 0: PROGRAM Warten;
[000] 1:

```

```

[E4B] 2: VAR DosZeit : ARRAY [0..3] OF BYTE ABSOLUTE
      $40:$6C;
[V24] 3:   Sekunden: INTEGER;
[000] 4:
[AP9] 5: PROCEDURE Warte (Wartezeit: REAL);
[YD3] 6: VAR i,j: REAL;
[B21] 7: BEGIN
[OF8] 8:   i := DosZeit[0] + DosZeit[1]*256
[UBA] 9:     + DosZeit[2]*65536.0 + DosZeit[3]*16
      777216.0;
[ZR1] 10:  j := i;
[8X6] 11:  WHILE (j < (i+Wartezeit)) DO
[1F8] 12:    j := DosZeit[0] + DosZeit[1]*256
[UBA] 13:      + DosZeit[2]*65536.0 + DosZeit[3]*
      16777216.0;
[TM0] 14:  END;
[000] 15:
[B21] 16: BEGIN
[NF2] 17:   ClrScr;
[DW3] 18:   Write ('Zeit? ');
[H65] 19:   ReadLn (Sekunden);
[NKC] 20:   WriteLn ('Ich warte ',Sekunden,' Sekunden
      ');
[SY5] 21:   Warte (Sekunden*18.2);
[C09] 22:   WriteLn ('Ich bin wieder da!');
[5L0] 23:   END.

```

Listing 4. Turbo Pascal wartet durch Lesen des MS-DOS-Zeitählers.

```

[C73] 1 REM ZEIT.BAS
[912] 10 T = TIMER
[0Z3] 20 FOR I% = 1 TO 10000
[YG1] 30 NEXT
[ZBB] 40 PRINT USING "##.##";TIMER-T;: PRINT " Sekunden"
[791] 50 END

```

Listing 5. Zeitmessung in GW-Basic – wie lange dauert die For-Schleife?

```

[XA5] 0: PROGRAM ZeitMessen;
[000] 1:
[315] 2: VAR StartTime: REAL;
[XTB] 3:   DosTime : ARRAY [0..3] OF BYTE ABSOLUT
      E $40:$6C;
[W63] 4:   i : INTEGER;
[000] 5:
[X46] 6: PROCEDURE StartTimer;
[B21] 7: BEGIN
[S6B] 8:   StartTime := DosTime[0] + DosTime[1]*256
[V8A] 9:     + DosTime[2]*65536.0 + DosTi
      me[3]*16777216.0;
[TM0] 10:   END;
[000] 11:
[WK6] 12: FUNCTION ReadTimer: REAL;
[TB2] 13: VAR x: REAL;
[B21] 14: BEGIN
[AD8] 15:   x := DosTime[0] + DosTime[1]*256
[V8A] 16:     + DosTime[2]*65536.0 + DosTime[3]*16
      777216.0;
[MK7] 17:   ReadTimer := x - StartTime;
[TM0] 18:   END;
[000] 19:
[B21] 20: BEGIN
[V93] 21:   StartTimer;
[NC4] 22:   FOR i := 1 TO 32767 DO
[V10] 23:     ;
[306] 24:   WriteLn (ReadTimer);
[5L0] 25:   END.

```

Listing 6. Auch in Turbo Pascal kann Zeit einfach gemessen werden.

```

[EH3] 1 REM SPRITE.BAS
[AP2] 10 DEFINT A-Z
[TRC] 20 KOMM$=CHR$(0)+"H"+CHR$(0)+"M"+CHR$(0)+"K"+CHR$(
      0)+"P"+" "+CHR$(13)
[GD8] 30 SP=79: X=12: OX=X: Y=39: OY=Y: G=1: Z=G
[AM4] 40 SCREEN 0,1: WIDTH 80
[XH4] 50 ON TIMER(1) GOSUB 1000
[8BB] 60 C=SCREEN(X+1,Y+1): LOCATE X+1,Y+1,0: PRINT CHR
      $(SP);
[D82] 70 TIMER ON
[D52] 80 I$=INKEY$
[SF3] 90 IF I$="" THEN 80
[C94] 100 I=INSTR(KOMM$,I$)
[1D3] 110 IF I=0 THEN 80
[VT9] 120 ON I GOTO 130,80,140,80,150,80,160,80,170,180
      ,190,200
[Y33] 130 D=1: GOTO 80
[743] 140 D=2: GOTO 80
[G43] 150 D=3: GOTO 80
[T43] 160 D=4: GOTO 80
[PP3] 170 G=G+1: GOTO 80

```



```

[T24] 180 IF G>1 THEN G=G-1
[X22] 181 GOTO 80
[653] 190 D=5: GOTO 80
[CS2] 200 TIMER OFF
[DN1] 210 END
[6G6] 1000 Z=Z-1: IF Z>0 THEN RETURN
[6S5] 1010 Z=G: IF D>5 THEN RETURN
[1X6] 1020 LOCATE X+1,Y+1: PRINT CHR$(C);
[2B7] 1030 ON D GOTO 1040,1050,1060,1070,1100
[3V5] 1040 X=(X+23) MOD 24: GOTO 1080
[3V5] 1050 Y=(Y+1) MOD 80: GOTO 1080
[TW5] 1060 Y=(Y+79) MOD 80: GOTO 1080
[DF3] 1070 X=(X+1) MOD 24
[1FB] 1080 C=SCREEN(X+1,Y+1): LOCATE X+1,Y+1: PRINT CHR
$(SP);
[E22] 1090 RETURN
[2TE] 1100 T=T XOR 1: LOCATE X,Y: IF T=0 THEN PRINT CHR
$(SP); ELSE PRINT " ";
[S12] 1110 RETURN

```

Listing 7. Eine Sprite-Routine in GW-Basic zeigt, was mit »On Timer« möglich ist

```

[Z65] 0: PROGRAM SpriteDemo;
[000] 1:
[NR5] 2: CONST Zeichen :BYTE = 2;
[5P5] 3: Attribut :BYTE = 7;
[DB4] 4: x :INTEGER = 39;
[U94] 5: y :INTEGER = 12;
[8W4] 6: OldX :INTEGER = 39;
[SU4] 7: OldY :INTEGER = 12;
[7W5] 8: Zaehler :INTEGER = 3;
[F25] 9: Geschwind:INTEGER = 3;
[MC5] 10: OldChar :BYTE = 0;
[UH5] 11: OldAttr :BYTE = 0;
[X54] 12: Dir :CHAR = #0;
[000] 13:
[UL9] 14: VAR VektorOff: INTEGER ABSOLUTE $0:$70;
[1E8] 15: VektorSeg: INTEGER ABSOLUTE $0:$72;
[5PC] 16: Screen : ARRAY [0..24,0..79,0..1] OF
BYTE ABSOLUTE $B800:0;
[DS2] 17: OldVekOff,
[5Y4] 18: OldVekSeg: INTEGER;
[FR2] 19: c : CHAR;
[000] 20:
[5X5] 21: PROCEDURE MoveSprite;
[B21] 22: BEGIN
[Y9C] 23: { rette alle Register außer AX, DX und DS
}
[000] 24:
[D35] 25: INLINE ($53/$51/$56/$57/$06);
[000] 26:
[N66] 27: Zaehler := Zaehler-1;
[9K5] 28: IF (Zaehler=0) THEN BEGIN
[XK6] 29: Zaehler := Geschwind;
[TB2] 30: CASE Dir OF
[PN4] 31: 'O': y := (y+24) MOD 25;
[5M4] 32: 'U': y := (y+1) MOD 25;
[8R4] 33: 'L': x := (x+79) MOD 80;
[NL4] 34: 'R': x := (x+1) MOD 80;
[000] 35: END;
[9B5] 36: IF (Dir<>' ') THEN BEGIN
[000] 37:
[919] 38: Screen[OldY,OldX,0] := OldChar;
[M59] 39: Screen[OldY,OldX,1] := OldAttr;
[LX6] 40: OldChar := Screen[Y,X,0];
[037] 41: OldAttr := Screen[Y,X,1];
[S27] 42: Screen[Y,X,0] := Zeichen;
[EB7] 43: Screen[Y,X,1] := Attribut;
[000] 44:
[165] 45: OldX := X; OldY := Y;
[DD0] 46: END
[ARO] 47: ELSE
[B8A] 48: Screen[Y,X,1] := Screen[Y,X,1] XOR $7
F;
[000] 49: END;
[000] 50:
[D8D] 51: { restoriere Register und führe IRET aus
}
[CA8] 52: INLINE ($07/$5F/$5E/$59/$5B/$8B/$E5/$5D/$
CF);
[000] 53: END;
[000] 54:
[000] 55:
[B21] 56: BEGIN
[LX6] 57: OldChar := Screen[Y,X,0];
[037] 58: OldAttr := Screen[Y,X,1];
[S27] 59: Screen[Y,X,0] := Zeichen;
[EB7] 60: Screen[Y,X,1] := Attribut;
[E56] 61: Geschwind := Zaehler;
[000] 62:
[1Z6] 63: OldVekOff := VektorOff;
[F07] 64: OldVekSeg := VektorSeg;
[000] 65:
[SD7] 66: { Interrupts sperren }
[FH2] 67: INLINE ($FA);
[AF8] 68: VektorOff := Ofs(MoveSprite);

```

```

[AR4] 69: VektorSeg := CSeg;
[8S7] 70: { Interrupts freigeben }
[KH2] 71: INLINE ($FB);
[000] 72:
[XG1] 73: REPEAT
[WR2] 74: Read (KBD,c);
[AR1] 75: CASE c OF
[828] 76: '-': Geschwind := Geschwind+1;
[T66] 77: '+': IF (Geschwind>1) THEN
[A97] 78: Geschwind := Geschwind-1;
[8S1] 79:
#27: BEGIN
[WR2] 80: Read (KBD,c);
[AR1] 81: CASE c OF
[453] 82: 'H': Dir := 'O';
[063] 83: 'M': Dir := 'R';
[S43] 84: 'K': Dir := 'L';
[S63] 85: 'P': Dir := 'U';
[000] 86: END;
[000] 87: END;
[UT2] 88: ' ': Dir := ' ';
[213] 89: #13: Dir := #13;
[000] 90: END;
[V13] 91: UNTIL (Dir=#13);
[000] 92:
[SA2] 93: INLINE($FA);
[347] 94: VektorOff := OldVekOff;
[H57] 95: VektorSeg := OldVekSeg;
[UA2] 96: INLINE($FB);
[000] 97:
[5L0] 98: END.

```

Listing 8. Die Routine »MoveSprite« erzeugt ein Zeichen auf dem Bildschirm, das sich automatisch bewegt

```

[BM3] 1 REM SOUND.BAS
[C53] 10 INPUT "J? ",J
[NN3] 20 IF J=0 THEN STOP
[BE4] 30 OUT 97,INP(97) OR 2
[183] 40 FOR K=0 TO J
[0H1] 50 NEXT
[MX4] 60 OUT 97,INP(97) AND &HFD
[7M4] 70 IF INKEY$<>" " THEN 10
[L22] 80 GOTO 30

```

Listing 9. Tonerzeugung in GW-Basic durch schnelles Ein- und Ausschalten des Lautsprechers

```

[T13] 0: (*$R-,V-,C-,U-*)
[ZV4] 1: PROGRAM Klavier;
[000] 2:
[000] 3:
[WUH] 4: CONST TimerCtrl = 67; { Steuerport f
ür den Timer 8253 }
[15C] 5: TimerData = 66; { Datenport -
----- " ----- }
[YM6] 6: SysCtrl = 97; { Schalterport
für den Lautsprecher }
[R7H] 7: IntCtrl = $20; { EOI-Port für
den Interrupt-Controller }
[XAE] 8: KbData = $60; { Datenport fü
r die Tastatur }
[05F] 9: KbCtrl = $61; { Steuerport f
ür die Tastatur }
[AB3] 10: c :BYTE = 0;
[RB3] 11: p :BYTE = 0;
[000] 12:
[000] 13:
[NB4] 14: VAR OldC : BYTE;
[FEB] 15: VektorOff : INTEGER ABSOLUTE $0:$24;
[MF8] 16: VektorSeg : INTEGER ABSOLUTE $0:$26;
[DS2] 17: OldVekOff,
[AT2] 18: OldVekSeg,
[KX4] 19: Octave : INTEGER;
[FME] 20: Regs : RECORD ax,bx,cx,dx,bp,si
,di,ds,es,fi: INTEGER;
[000] 21:
[000] 22: END;
[DE4] 23: PROCEDURE GetCh;
[B21] 24: BEGIN
[8V7] 25: { rette alle Register }
[427] 26: INLINE ($50/$51/$52/$53/$56/$57/$1E/$06);
[000] 27:
[UR4] 28: c := Port[KbData];
[000] 29:
[6V4] 30: p := Port[KbCtrl];
[236] 31: Port[KbCtrl] := p OR $80;
[T56] 32: Port[KbCtrl] := p AND $7F;
[000] 33:
[GG5] 34: Port[IntCtrl] := $20;
[000] 35:
[D8D] 36: { restoriere Register und führe IRET aus
}
[ZGA] 37: INLINE ($07/$1F/$5F/$5E/$5B/$5A/$59/$58/$
8B/$E5/$5D/$CF);
[000] 38: END;
[000] 39:

```



```

[000] 40:
[ED4] 41: PROCEDURE Play;
[000] 42:
[XG9] 43: CONST Notes: ARRAY[0..6, 0..11] OF INTEGER
=
[000] 44:
[UX9] 45: { C C# D D# E F F#
G G# A A# B }
[810] 46: (
[7EC] 47: (18357,17292,16345,15297,14551,13715,12969,
12175,11473,10847,10286, 9701),
[1ZB] 48: ( 9108, 8584, 8117, 7649, 7231, 6818, 6450,
6088, 5736, 5424, 5121, 4831),
[8UB] 49: ( 4554, 4292, 4058, 3824, 3616, 3409, 3225,
3044, 2868, 2712, 2560, 2415),
[ZPB] 50: ( 2277, 2146, 2029, 1912, 1808, 1705, 1612,
1522, 1434, 1356, 1280, 1208),
[X5B] 51: ( 1139, 1073, 1015, 956, 904, 852, 806,
761, 717, 678, 640, 604),
[SZA] 52: ( 569, 537, 507, 478, 452, 426, 403,
380, 359, 339, 320, 302),
[DVA] 53: ( 285, 268, 254, 239, 226, 213, 202,
190, 179, 169, 160, 151)
[Z40] 54: );
[000] 55:
[UK3] 56: VAR i: INTEGER;
[000] 57:
[B21] 58: BEGIN
[2W3] 59: WHILE TRUE DO BEGIN
[NE5] 60: REPEAT UNTIL (c<>OldC);
[AR1] 61: CASE c OF
[H92] 62: 31 : i := 0;
[1A2] 63: 18 : i := 1;
[V92] 64: 32 : i := 2;
[BA2] 65: 19 : i := 3;
[5A2] 66: 33 : i := 4;
[BA2] 67: 34 : i := 5;
[8A2] 68: 21 : i := 6;
[NA2] 69: 35 : i := 7;
[KA2] 70: 22 : i := 8;
[ZA2] 71: 36 : i := 9;
[3L2] 72: 23 : i := 10;
[HL2] 73: 37 : i := 11;
[PP5] 74: 42 : IF (Octave>0) THEN

```

```

[VB5] 75: Octave := Octave-1;
[SR5] 76: 54 : IF (Octave<6) THEN
[TB5] 77: Octave := Octave+1;
[9U1] 78: 28 : Exit;
[HVD] 79: 146..165: Port[SysCtrl] := Port[SysCtrl] AND $FC;
[TM0] 80: END;
[000] 81:
[ZY1] 82: OldC := c;
[TB6] 83: IF (c>=18) AND (c<=37) THEN BEGIN
[GZ6] 84: Port[TimerCtrl] := $B6;
[TZB] 85: Port[TimerData] := Lo(Notes[Octave,i]);
[BYB] 86: Port[TimerData] := Hi(Notes[Octave,i]);
[3FA] 87: Port[SysCtrl] := Port[SysCtrl] OR 3;
[TM0] 88: END;
[TM0] 89: END;
[TM0] 90: END;
[000] 91:
[000] 92:
[B21] 93: BEGIN
[ZY1] 94: OldC := c;
[1Z6] 95: OldVekOff := VektorOff;
[F07] 96: OldVekSeg := VektorSeg;
[000] 97:
[SA2] 98: INLINE($FA);
[GU6] 99: VektorOff := Ofs(GetCh);
[AR4] 100: VektorSeg := CSeg;
[UA2] 101: INLINE($FB);
[000] 102:
[3W2] 103: Octave := 3;
[EEB] 104: Port[SysCtrl] := Port[SysCtrl] AND $FC;
[PA1] 105: Play;
[000] 106:
[EEB] 107: Port[SysCtrl] := Port[SysCtrl] AND $FC;
[SA2] 108: INLINE($FA);
[347] 109: VektorOff := OldVekOff;
[H57] 110: VektorSeg := OldVekSeg;
[UA2] 111: INLINE($FB);
[5L0] 112: END.

```

Listing 10. Der PC wird zum musikalischen Instrument – das Programm »Klavier«

TURBO PASCAL SOURCE CODE

NEU:
Version für Turbo Pascal 4.0

TURBO SCREEN

TURBO SCREEN+

TURBO DATA

TURBO GEN 1

TURBO FORMATTER

jetzt
mit eigenem
Isam-System

alle fünf
zusammen nur DM 999,-

Tests in: Computer Persönlich 8/88
DOS 12/87, PASCAL 3/88
PC-WELT 2/88

Einzelpakete jeweils DM 224,-
Scheck oder Nachnahme / Infodiskette anfordern
TURBO PASCAL ist Warenzeichen von Borland Int.

Zippelhaus 4
2 Hamburg 11

Telefon:
040/337478



COMPUTER-INFORMATIONSDIENST GMBH

Copro

Leisten Sie sich Urlaub und Computer!

AT-286 Komplettsystem 2970,-

512 KByte RAM, 6/10 MHz Takt umschaltbar,
1,2 MByte Diskettenlaufwerk, 20 MByte Fest-
platte, Hercules kompatible Grafikkarte, Parallele
Druckerschnittstelle, batteriegepufferte Echt-
zeituhr/Kalender, Deutsche DIN-Tastatur mit
abgesetztem Cursorblock, 14" Monitor bernstein,
deutsche Bedienungshandbücher, 1 volles
Jahr Garantie

AT-286/60 CAD

wie AT-286/30, jedoch: 12 MHz Takt ohne wait-
state, Norton-Faktor 13,7, Landmark-Test
16,1 MHz, 80 nsec-RAM's, 60 MByte Fest-
platte 5450,-

AT-286/20 Portable, 640KB RAM,
6/12 MHz Takt umschaltbar, 1 wait-state, 1,2
MByte Diskettenlaufwerk, 20MByte Festplatte,
Hercules-kompatible Grafikkarte, 7" Monitor
Flatscreen bernstein. 4570,-

Farbgrafik:

EGA-Karte 640 x 350 Pkt., mit AOC EGA-Monitor: Aufpreis 690,-
EGA-Karte »Genoa«, 640 x 480 Pkt., mit Monitor Fujitsu Multisync: Aufpreis 1090,-
EGA-Karte »Ati EGA Wonder«, 800x600 Pkt., m. Mon. Eizo Flexscan 8060S: Aufpreis 1690,-
Selbstverständlich führen wir weitere AT- und XT-Modelle. Fordern Sie unser kostenloses Info an.
Viele weitere Produkte auf Anfrage. Händleranfragen sind uns stets willkommen. Schreiben Sie
uns oder rufen Sie einfach an!

Wir finden die optimale Lösung für Ihre Anwendung

Copro · Berger Straße 15 · 6369 Niederdorfelden
☎ 06101-3890 · Bei Bestellung bitte »DE« angeben



Zeichensatz

– selbst programmiert

Der selbst programmierte Zeichensatz versetzt Sie in die Lage, mit Ihrem Matrixdrucker neue Schriften zu kreieren. Voraussetzung ist die Kompatibilität Ihres Druckers zum Epson-Standard.

Kennt Ihr Drucker den IBM-Grafikzeichensatz noch nicht? Ab sofort kann mit dem hier vorgestellten Programm auch er diesen Grafikzeichensatz drucken. Das Programm ist dabei in der Lage, eine Textdatei genauso wiederzugeben, wie sie auf dem Bildschirm dargestellt wird. Eine Ausnahme bilden allerdings die Steuerzeichen, die natürlich nicht gedruckt werden können.

Wollten Sie zum Beispiel das Zeichen »§« ausdrucken, so geht das bisher nur, wenn Sie den Drucker vorher auf den deutschen Zeichensatz einstellen. Im Text müssen Sie unter Umständen das Zeichen »|« eintippen, um das Paragraph-Zeichen auf dem Drucker zu erhalten. Eine Lösung, die sicher nicht komfortabel ist.

Das Programm bewirkt nun, ohne jegliche Druckerinstallation alle Sonderzeichen gleichzeitig ausdrucken zu können. Folgende Druckfunktionen werden dabei vom Ausgabe-Programm unterstützt:

Breitschrift – »Esc W (n)«
 Fettdruck ein – »Esc E« oder »Esc G«
 Fettdruck aus – »Esc F« oder »Esc H«
 Hochstellen – »Esc S (n)«
 Tiefstellen – »Esc T«
 Unterstreichen – »Esc - (n)«

Beim Hoch- und Tiefstellen benutzt das Programm den druckereigenen Zeichensatz. Zusätzlich ist es in der Lage die Druckerbefehle zur Einstellung des Zeilenabstands zu benutzen.

Das Programmpaket besteht aus zwei Teilen mit den Namen »prn_def« (Listing 1) und »prn_8« (Listing 3) (beziehungsweise »prn_9«, Listing 4). Für die Definition des Zeichensatzes ist das Programm »prn_def« verantwortlich, wobei für »normale« Zeichen ein Raster der Größe 12x18 Punkte zur Verfügung steht. Grafikzeichen lassen

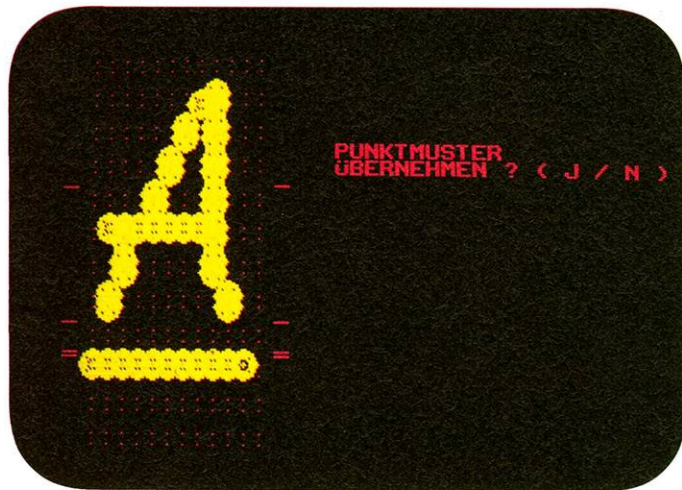


Bild 1. Der Zeichensatz-Editor funktioniert nur mit CGA-Farbkarte

sich mit einer Auflösung von 12x24 Punkten ausgeben (Bild 1).

Vor dem Programmstart aktivieren Sie mit dem MS-DOS-Programm »Graftab« von der Systemdiskette die Zeichen des erweiterten ASCII-Zeichensatzes. Außerdem müssen sich die Turbo Pascal-Dateien »Graph.p« und »Graph.bin« auf der aktuellen Diskette befinden. Beim erstmaligen Start des Programms legen Sie unbedingt eine leere Datei mit dem Namen »const.dta« ins aktuelle Laufwerk, diese Datei beherbergt später Ihre selbstdefinierten Zeichen. Listing 2 enthält einen Zeichensatz (Bild 2), der schon eine neue Schrift umfaßt. Diesen können Sie als »leere« Datei für Ihre eigenen Kreationen benutzen.

Im Zeichensatzeditor wird der Cursor mit Hilfe der Cursor-Steuertasten bewegt. Mit der Leertaste oder mit [+] setzen Sie Punkte. Mit [-] löschen Sie sie wieder. Haben Sie die Tasten [Z] und [+] gedrückt, zeichnet der Cursor eine Linie, egal wohin Sie ihn bewegen. Eine »dauerhafte« Löschfunktion wird durch die Tasten [Z] und [-] eingeleitet. Das gesamte Punktraster läßt sich durch die Taste [X] löschen. [Y] fungiert als Undo-Befehl, der das ursprüngliche Punktraster wieder herstellt.

Betätigen Sie die V-Taste, ordnet das Programm einem Zeichen einen anderen ASCII-Wert zu. Haben Sie zum

Mit diesem Programm kann man auf jedem EPSON kompatiblen Matrixdrucker selbstdefinierte Schönschriften ausdrucken. Ein Drucker den der IBM Grafikzeichensatz nicht kennt, kann diese ab sofort darstellen. D.h. das Programm ist in der Lage, eine Textdatei genauso wiederzugeben wie sie auf dem Bildschirm dargestellt wird. Eine Ausnahme bilden die wenigen Steuerzeichen. Wollte man bisher z.B. das Zeichen '8' ausdrucken, so ging dies nur indem man vorher den Drucker auf den Deutschen Zeichensatz einstellte, und dann mußte man im Text unter Umständen noch ']' eintippen um das Zeichen zu generieren. Gleichzeitig kann der 'Schönschriftendrucker', natürlich auch ohne irgendeine Installation, alle internationalen Sonderzeichen ausdrucken.

Der 'Schönschriftendrucker' kennt Breitschrift (ESC W (n)), er kennt Fettdruck (ESC E oder G zum einschalten und F oder H zum ausschalten), zum Hoch- und Tiefstellen von Zeichen (ESC S (n) bzw ESC T) verwendet das Programm den druckereigenen Zeichensatz. Der Schönschriftendrucker kann unterstreichen (ESC - (n)) und kennt die Befehle zur Einstellung des Zeilenabstands.

Zum Programm selbst

'Schönschriftendrucker' besteht aus zwei Teilen:

Bild 2. Ein Probeausdruck mit dem Zeichensatz der Databox-Diskette

Beispiel schon ein »kleines a« definiert und wollen daraus ein »kleines d« machen, so rufen Sie das »kleine a« auf, drücken die Tasten [V] und [d] und wandeln dann das »a« in ein »d« um.

Mit der Taste [D] wird ein frei definierbarer Teil des Zeichensatzes ausgedruckt, allerdings maximal 96 Zeichen. Die Return-Taste schließt den Aufbau eines Buchstabens ab. [Q] beendet das Programm und speichert die Änderungen in der Datei »const.dta« ab.

Nach dem Speichern stehen zwei solcher Dateien auf der Diskette: »const.dta« und »const.alt«. Sind Sie mit Ihrem geänderten Zeichensatz nicht zufrieden, steht Ihnen immer noch die letzte Version in der Datei »const.alt« zur Verfügung. Das Programm legt die Daten des Zeichensatzes in dieser Datei in Form von »Konstanten-Deklarationen« ab, die dann als Include-Datei gelesen wird. Diese Form der Datenspeicherung wurde aus einem bestimmten Grund gewählt. Sie weist nämlich einen erheblichen Geschwindigkeitsvorteil gegen-

über dem komponentenweisen Lesen der Daten auf. Da die Datei ein Teil des Programms ist, erkennt der Zeichengenerator die Änderungen in der Datei »const.dta« erst nach einer erneuten Compilierung. Aus diesem Grund ist es nicht sinnvoll, den Zeichengenerator als Com-Datei anzulegen.

Ist Ihr spezieller Zeichensatz fertiggestellt, können Sie jede beliebige ASCII-Datei wahlweise mit den Programmen »prn_8« oder »prn_9« zu Papier bringen. Das Programm »prn_8« eignet sich für alle Epson-kompatiblen Matrixdrucker.

Die zweite Version des Druckprogrammes (»prn_9«) ist für Drucker gedacht, die den 9-Nadel-Grafik-Modus mit dem Befehl »Esc ↑« kennen. Diese Version ist im Vergleich zu »prn_8« etwas schneller, da sie kürzer ist. Informieren Sie sich also zuerst in Ihrem Druckerhandbuch, welcher Modus in Frage kommt, bevor Sie eines der beiden Druckprogramme abtippen.

(Günter Fritz/sp)

```
[FOD] 0: PROGRAM NLQMATRIX ;{*** 19.08.87 ** Günter
      Fritz ***}
[000] 1:
[TX2] 2: ($I GRAPH.P )
[5X3] 3: ($I CONST.DTA )
[000] 4:
[TB5] 5: TYPE NAME = STRING [ 40 ];
[BM7] 6: FELD = ARRAY [ 1..12, 1..255 ] OF BYTE
      ;
[000] 7:
[8B7] 8: VAR EING, ZEICHEN, ZEICHNE : CHAR;
[DZC] 9: ALAUF, BLAUF, CLAUF, XORT, YORT, NUMLOC
      K, ASCII, BIN : BYTE;
[RPB] 10: OBENPUNKTE_1, OBENPUNKTE_2, UNTENPUNKTE
      _ : FELD;
[PR2] 11: DATEI : TEXT;
[HA9] 12: AKTPUNKT : ARRAY [ 1..12, 1..24 ] OF BO
      OLEAN;
[UHF] 13: UNVERAENDERT, CHUNVERAENDERT, BILD, ZEI
      CHNE_PLUS, ZEICHNE_MINUS : BOOLEAN;
[000] 14:
[PS9] 15: LABEL ENDE, NeuerBuchstabe, NEUSTART,
[VO6] 16: ZEICHNEPLUS, ZEICHNEMINUS;
[000] 17:
[PIA] 18: FUNCTION XADDR ( XORT : INTEGER ) : INTEGER
      ;
[ZP6] 19: BEGIN XADDR := ( XORT * 8 ) + 8 END;
[000] 20:
[U1A] 21: FUNCTION YADDR ( YORT : INTEGER ) : INTEGER
      ;
[B21] 22: BEGIN
[DR9] 23: IF ODD ( YORT ) THEN YADDR := ( ( 25 - YOR
      T ) * 8 ) - 3
[ZT5] 24: ELSE YADDR := ( ( 25 - YORT ) * 8 )
[TH0] 25: END;
[000] 26:
[CL4] 27: PROCEDURE LESEDATEI;
[B21] 28: BEGIN
[RR4] 29: UNVERAENDERT := TRUE;
[CV5] 30: CHUNVERAENDERT := TRUE;
[UC4] 31: FOR ALAUF := 1 TO 255 DO
[CK4] 32: FOR BLAUF := 1 TO 12 DO
[B21] 33: BEGIN
[FED] 34: OBENPUNKTE_1 [ BLAUF, ALAUF ] :
      = OBENPUNKTE1 [ ALAUF, BLAUF ];
[RED] 35: OBENPUNKTE_2 [ BLAUF, ALAUF ] :
      = OBENPUNKTE2 [ ALAUF, BLAUF ];
[WRD] 36: UNTENPUNKTE [ BLAUF, ALAUF ] :
      = UNTENPUNKTE [ ALAUF, BLAUF ];
[EB3] 37: END; ( BLAUF )
[Z23] 38: END; ( LESEDATEI )
[000] 39:
[4U8] 40: PROCEDURE PUNKT ( X, Y, FARBE : BYTE );
[XN3] 41: VAR ALAUF : BYTE;
[B21] 42: BEGIN
[Y64] 43: FOR ALAUF := 1 TO 7 DO
[9S5] 44: CIRCLE ( X, Y, ALAUF, FARBE );
[TH0] 45: END;
[000] 46:
[6E6] 47: PROCEDURE BILDSCHIRMAUFBAU;
[7TB] 48: VAR XVAR1, XVAR2, YVAR1, YVAR2 : BYTE;
[B21] 49: BEGIN
[TH2] 50: IF BILD THEN
```

```
[B21] 51: BEGIN
[5D3] 52: XVAR1 := XORT - 1;
[666] 53: IF XVAR1 = 0 THEN XVAR1 := 1;
[WC3] 54: XVAR2 := XORT + 1;
[CV6] 55: IF XVAR2 = 13 THEN XVAR2 := 12;
[8D3] 56: YVAR1 := YORT - 1;
[D66] 57: IF YVAR1 = 0 THEN YVAR1 := 1;
[ZC3] 58: YVAR2 := YORT + 1;
[PF6] 59: IF YVAR2 = 25 THEN YVAR2 := 24
[DD0] 60: END
[Y11] 61: ELSE BEGIN
[DW1] 62: XVAR1 := 1;
[332] 63: XVAR2 := 12;
[EW1] 64: YVAR1 := 1;
[C32] 65: YVAR2 := 24;
[OF6] 66: GOTOXY ( 1, 9 ); WRITE ( '-' );
[FS6] 67: GOTOXY ( 1, 17 ); WRITE ( '-' );
[ZT6] 68: GOTOXY ( 1, 19 ); WRITE ( '=' );
[HS6] 69: GOTOXY ( 15, 9 ); WRITE ( '-' );
[Z07] 70: GOTOXY ( 15, 17 ); WRITE ( '-' );
[U27] 71: GOTOXY ( 15, 19 ); WRITE ( '=' );
[G33] 72: GOTOXY ( 20, 1 );
[1C9] 73: WRITE ( '-', CHR ( ASCII ), ' -
      ( ', ASCII, ' ' ) );
[TH0] 74: END;
[ZM7] 75: FOR ALAUF := YVAR2 DOWTO YVAR1 DO
[5N6] 76: FOR BLAUF := XVAR1 TO XVAR2 DO
[B21] 77: BEGIN
[767] 78: IF AKTPUNKT [ BLAUF, ALAUF ] THEN
[B21] 79: BEGIN
[KO9] 80: PUNKT ( XADDR ( BLAUF ), YADDR (
      ALAUF ), 3 );
[S72] 81: END; ( IF )
[PW7] 82: PLOT ( XADDR ( BLAUF ), YADDR ( ALA
      UF ), 2 );
[EB3] 83: END; ( BLAUF )
[CN7] 84: IF AKTPUNKT [ XORT, YORT ] THEN
[2E9] 85: CIRCLE ( XADDR ( XORT ), YADDR ( YORT ),
      2, 0 )
[GU9] 86: ELSE CIRCLE ( XADDR ( XORT ), YADDR ( YORT
      ), 2, 2 );
[1N2] 87: BILD := TRUE;
[SV5] 88: END; ( BILDSCHIRMAUFBAU )
[000] 89:
[624] 90: PROCEDURE UEBERNEHME;
[B21] 91: BEGIN
[EW5] 92: IF NOT CHUNVERAENDERT THEN
[B21] 93: BEGIN
[D53] 94: GOTOXY ( 19, 7 );
[526] 95: WRITELN ( 'PUNKTMUSTER' );
[L53] 96: GOTOXY ( 19, 8 );
[SV7] 97: WRITELN ( 'ÜBERNEHMEN ? ( J / N ) ' )
      ;
[XG1] 98: REPEAT
[R54] 99: READ ( KBD, ZEICHEN );
[MLD] 100: UNTIL ( ZEICHEN IN [ 'J', 'j', 'N', '
      n' ] ) OR ( ORD ( ZEICHEN ) = 26 );
[LE5] 101: IF ZEICHEN IN [ 'J', 'j' ]
      THEN BEGIN
[S35] 103: UNVERAENDERT := FALSE;
[KM5] 104: FOR ALAUF := 1 TO 12 DO BEGIN
[177] 105: OBENPUNKTE_1 [ ALAUF, ASCII
```



```

[Z57] 106:          OBENPUNKTE_2 [ ALAUF, ASCII
[AB7] 107:          ] := 0;          UNTENPUNKTE_ [ ALAUF, ASCII
[9F1] 108:          ] := 0;
[055] 109:          BIN := 1;
[SK6] 110:          FOR BLAUF := 1 TO 4 DO BEGI
N
[SK6] 110:          IF AKTPUNKT [ ALAUF, 2 *
[110] 111:          THEN UNTENPUNKTE_ [ AL
[111] 112:          AUF, ASCII ]
[HK9] 112:          := UNTENPUNKTE_ [
[112] 113:          ALAUF, ASCII ] + 16 * BIN;
[E57] 113:          IF AKTPUNKT [ ALAUF, 2 *
[113] 114:          BLAUF - 1 ]
[114] 115:          THEN UNTENPUNKTE_ [ AL
[115] 116:          AUF, ASCII ]
[116] 117:          := UNTENPUNKTE_ [
[117] 118:          ALAUF, ASCII ] + BIN;
[833] 118:          BIN := 2 * BIN;
[265] 119:          IF BLAUF = 4 THEN BIN :=
1
[EB3] 119:          END; ( BLAUF )
[9F1] 120:          BIN := 1;
[FN5] 120:          FOR BLAUF := 5 TO 12 DO BEG
IN
[SK6] 121:          IF AKTPUNKT [ ALAUF, 2 *
[121] 122:          BLAUF ]
[CM7] 122:          THEN OBENPUNKTE_1 [ AL
[122] 123:          AUF, ASCII ]
[749] 123:          := OBENPUNKTE_1 [
[123] 124:          ALAUF, ASCII ] + BIN;
[E57] 124:          IF AKTPUNKT [ ALAUF, 2 *
[124] 125:          BLAUF - 1 ]
[125] 126:          THEN OBENPUNKTE_2 [ AL
[126] 127:          AUF, ASCII ]
[E49] 127:          := OBENPUNKTE_2 [
[127] 128:          ALAUF, ASCII ] + BIN;
[833] 128:          BIN := 2 * BIN;
[G65] 128:          IF BLAUF = 12 THEN BIN :
= 1
[EB3] 129:          END; ( BLAUF )
[EB3] 130:          END; ( ALAUF )
[Z44] 131:          END; ( IF ZEICHEN )
[636] 132:          END; ( IF CHUNVERAENDERT )
[L74] 133:          END; ( UEBERNEHME )
[000] 134:
[D24] 135:          PROCEDURE DRUCKE;
[OB5] 136:          VAR ASCII1, ASCII2 : BYTE;
[CS3] 137:          ANZAHL : INTEGER;
[E9C] 138:          PROCEDURE VORSCHUB ( ZAHL : BYTE ); ( IN
216tel '' )
[B21] 139:          BEGIN
[5SB] 140:          WRITE ( LST, CHR ( 27 ), CHR ( 74 ), CH
R ( ZAHL ), CHR ( 13 ) );
[140] 141:          END;
[B21] 142:          BEGIN
[022] 143:          ASCII1 := 0;
[622] 144:          ASCII2 := 0;
[DB3] 145:          GOTOXY ( 20, 25 );
[H77] 146:          WRITE ( 'Iter ASCII-Wert ? ' );
[BP2] 147:          READ ( ASCII1 );
[DB3] 148:          GOTOXY ( 20, 25 );
[ED4] 149:          WRITE ( ' ' );
[DB3] 150:          GOTOXY ( 20, 25 );
[FZ6] 151:          WRITE ( '2ter ASCII-Wert ? ' );
[GP2] 152:          READ ( ASCII2 );
[DB3] 153:          GOTOXY ( 20, 25 );
[ED4] 154:          WRITE ( ' ' );
[8P7] 155:          ANZAHL := 12 * ( ASCII2 - ASCII1 + 1 );
[V29] 156:          IF ( ASCII1 <> 0 ) AND ( ASCII2 <> 0 ) THEN
BEGIN
[RA3] 157:          VORSCHUB ( 3 );
[Z64] 158:          FOR CLAUF := 1 TO 4 DO
[B21] 159:          BEGIN
[159] 160:          WRITE ( LST, CHR ( 27 ), 'L',
[SE4] 161:          CHR ( ANZAHL MOD 256 ),
[9H5] 162:          CHR ( TRUNC ( ANZAHL / 256 )
));
[LR2] 163:          CASE CLAUF OF
[6P1] 164:          1 : BEGIN
[VL6] 165:          FOR ALAUF := ASCII1 TO ASCII
2 DO
[CK4] 166:          FOR BLAUF := 1 TO 12 DO
[F4B] 167:          WRITE ( LST, CHR ( OBE
NPUNKTE_1 [ BLAUF, ALAUF ] );
[FA3] 168:          VORSCHUB ( 1 );
[168] 169:          END;
[169] 170:          2 : BEGIN
[VL6] 171:          FOR ALAUF := ASCII1 TO ASCII
2 DO
[CK4] 172:          FOR BLAUF := 1 TO 12 DO
[F4B] 173:          WRITE ( LST, CHR ( OBE
NPUNKTE_2 [ BLAUF, ALAUF ] );

```

```

[G73] 174:          VORSCHUB ( 11 );
[174] 175:          END;
[8P1] 176:          3 : BEGIN
[VL6] 177:          FOR ALAUF := ASCII1 TO ASCII
2 DO
[CK4] 178:          FOR BLAUF := 1 TO 12 DO
[RRC] 179:          WRITE ( LST, CHR ( TRUN
C ( ( UNTENPUNKTE_ [ BLAUF, ALAUF ]
[179] 180:          AND 240 ) / 16 ) );
[FA3] 181:          VORSCHUB ( 1 );
[181] 182:          END;
[XP7] 183:          4 : FOR ALAUF := ASCII1 TO ASCII2
DO
[CK4] 184:          FOR BLAUF := 1 TO 12 DO
[67B] 185:          WRITE ( LST, CHR ( UNT
ENPUNKTE_ [ BLAUF, ALAUF ]
[185] 186:          AND 15 ) );
[2X2] 187:          END; ( CASE )
[EB3] 188:          END; ( CLAUF )
[HS3] 189:          END; ( IF ASCII )
[GV3] 190:          END; ( DRUCKE )
[000] 191:
[B21] 192:          BEGIN
[5R5] 193:          NUMLOCK := MEM[0:$417];
[U4A] 194:          MEM[0:$417] := MEM[0:$417] OR $20; ( NUM LO
CK EIN )
[213] 195:          BILD := FALSE;
[862] 196:          LESEDATEI;
[G85] 197:          NeuerBuchstabe;
[852] 198:          TEXTMODE;
[NZ1] 199:          CLRSCR;
[8ZN] 200:          WRITELN ( 'Geben Sie den zu definierenden
Buchstaben oder den ASCII- CODE ' );
[AVF] 201:          WRITELN ( 'des Zeichens angefuert von einem
'' # '' ein. ' );
[932] 202:          WRITELN;
[HVL] 203:          WRITELN ( 'Das Zeichen '' # '' ( - 35 - ),
sowie alle anderen Zeichen, die ');
[XZS] 204:          WRITELN ( 'nicht direkt über die Tastatur er
reichbar sind, können, auf diese ');
[2XR] 205:          WRITELN ( 'Weise ( oder mit Hilfe von Alt-T
aste und Ziffernblock ) erzeugt ');
[ME5] 206:          WRITELN ( 'werden. ' );
[R54] 207:          READ ( KBD, ZEICHEN );
[9R3] 208:          WRITE ( ZEICHEN );
[OW6] 209:          IF ZEICHEN = '#' THEN READ ( ASCII )
[S66] 210:          ELSE ASCII := ORD ( ZEICHEN );
[PX3] 211:          GRAPHCOLORMODE;
[PZ2] 212:          PALETTE ( 2 );
[562] 213:          NEUSTART;
[R35] 214:          ZEICHNE_PLUS := FALSE;
[NL5] 215:          ZEICHNE_MINUS := FALSE;
[KM5] 216:          FOR ALAUF := 1 TO 12 DO BEGIN
[9F1] 217:          BIN := 1;
[055] 218:          FOR BLAUF := 1 TO 4 DO BEGIN
[LVC] 219:          IF ( UNTENPUNKTE_ [ ALAUF, ASCII ] A
ND ( 16 * BIN ) = 16 * BIN
[BR0] 220:          THEN
[XY7] 221:          AKTPUNKT [ ALAUF, 2 * BLAUF ]
:= TRUE
[AR0] 222:          ELSE
[BR8] 223:          AKTPUNKT [ ALAUF, 2 * BLAUF ]
:= FALSE;
[FEB] 224:          IF ( UNTENPUNKTE_ [ ALAUF, ASCII ] A
ND BIN ) = BIN
[BR0] 225:          THEN
[XC8] 226:          AKTPUNKT [ ALAUF, 2 * BLAUF -
1 ] := TRUE
[AR0] 227:          ELSE
[VG8] 228:          AKTPUNKT [ ALAUF, 2 * BLAUF -
1 ] := FALSE;
[833] 229:          BIN := 2 * BIN;
[265] 230:          IF BLAUF = 4 THEN BIN := 1
[EB3] 231:          END; ( BLAUF )
[9F1] 232:          BIN := 1;
[FN5] 233:          FOR BLAUF := 5 TO 12 DO BEGIN
[7FB] 234:          IF ( OBENPUNKTE_1 [ ALAUF, ASCII ] A
ND BIN ) = BIN
[BR0] 235:          THEN
[XY7] 236:          AKTPUNKT [ ALAUF, 2 * BLAUF ]
:= TRUE
[AR0] 237:          ELSE
[BR8] 238:          AKTPUNKT [ ALAUF, 2 * BLAUF ]
:= FALSE;
[87B] 239:          IF ( OBENPUNKTE_2 [ ALAUF, ASCII ] A
ND BIN ) = BIN
[BR0] 240:          THEN
[XC8] 241:          AKTPUNKT [ ALAUF, 2 * BLAUF -
1 ] := TRUE
[AR0] 242:          ELSE
[VG8] 243:          AKTPUNKT [ ALAUF, 2 * BLAUF -
1 ] := FALSE;

```



```

[833] 244:      BIN := 2 * BIN;
[G65] 245:      IF BLAUF = 12 THEN BIN := 1
[EB3] 246:      END; ( BLAUF )
[EB3] 247:      END; ( ALAUF )
[000] 248:
[CV5] 249:      CHUNVERAENDERT := TRUE;
[MN2] 250:      CLEARSCREEN;
[213] 251:      BILD := FALSE;
[AV1] 252:      XORT := 1;
[F12] 253:      YORT := 24;
[023] 254:      BILDSCHIRMAUFBAU;
[0T2] 255:      WHILE 1 = 1 DO
[B21] 256:      BEGIN
[1A3] 257:      READ ( KBD, EING );
[WT4] 258:      IF ORD ( EING ) = 13 THEN
[B21] 259:      BEGIN
[KC2] 260:      UEBERNEHME;
[MN2] 261:      CLEARSCREEN;
[FG5] 262:      GOTO NeuerBuchstabe
[TM0] 263:      END;
[VC2] 264:      CASE EING OF
[YE3] 265:      'Z', 'z' : BEGIN
[WX9] 266:      IF ZEICHNE_PLUS THEN ZEICH
NE_PLUS := FALSE
[EHB] 267:      ELSE IF ZEICHNE_MINUS TH
EN ZEICHNE_MINUS := FALSE
[YY1] 268:      ELSE BEGIN
[G33] 269:      GOTOXY ( 20, 1 );
[LP5] 270:      WRITE ( 'ZEICHNE
( + / - ) ? ' );
[9E5] 271:      REPEAT READ ( KBD
, ZEICHNE )
[UA6] 272:      UNTIL ( ZEICHNE
IN [ '+', '-' ] )
[7Y4] 273:      OR ( ORD ( ZEIC
HNE ) = 13 );
[NY8] 274:      IF ZEICHNE = '+'
THEN ZEICHNE_PLUS := TRUE
[W5B] 275:      ELSE IF ZEICHNE
= '-' THEN ZEICHNE_MINUS := TRUE;
[G33] 276:      GOTOXY ( 20, 1 );
[3A4] 277:      WRITE ( '
' );
[G33] 278:      GOTOXY ( 20, 1 );
[DX2] 279:      END; ( ELSE )
[NO2] 280:      END; ( Z )
[6K3] 281:      'D', 'd' : DRUCKE;
[UD3] 282:      'V', 'v' : BEGIN
[R54] 283:      READ ( KBD, ZEICHEN );
[F34] 284:      IF ZEICHEN = '#' THEN
[B21] 285:      BEGIN
[G33] 286:      GOTOXY ( 20, 1 );
[3A4] 287:      WRITE ( '
' );
[G33] 288:      GOTOXY ( 20, 1 );
[RE3] 289:      WRITE ( '#', ' ' );
[DC2] 290:      READ ( ASCII )
[DD0] 291:      END
[S66] 292:      ELSE ASCII := ORD ( ZEICHE
N );
[BV5] 293:      CHUNVERAENDERT := FALSE;
[G33] 294:      GOTOXY ( 20, 1 );
[3A4] 295:      WRITE ( '
' );

```

```

[G33] 296:      GOTOXY ( 20, 1 );
[1C9] 297:      WRITE ( '-', CHR ( ASCII
), ' - ( ' , ASCII, ' ' );
[TH0] 298:      END;
[M49] 299:      'Q', 'q' : BEGIN UEBERNEHME; GOTO ENDE
END;
[TU5] 300:      'Y', 'y' : GOTO NEUSTART;
[CE3] 301:      'X', 'x' : BEGIN
[7K4] 302:      FOR ALAUF := 1 TO 12 DO
[RK4] 303:      FOR BLAUF := 1 TO 24 DO
[LW7] 304:      AKTPUNKT [ ALAUF, BL
AUF ] := FALSE;
[MN2] 305:      CLEARSCREEN;
[213] 306:      BILD := FALSE;
[5X3] 307:      BILDSCHIRMAUFBAU
[TM0] 308:      END;
[NV4] 309:      '+', ' ' : ZEICHNEPLUS;
[B21] 310:      BEGIN
[EC9] 311:      IF NOT AKTPUNKT [ XORT, YO
RT ] THEN BEGIN
[000] 312:
[BV5] 313:      CHUNVERAENDERT := FALSE;
[VC8] 314:      PUNKT ( XADDR ( XORT ),
YADDR ( YORT ), 3 );
[UW6] 315:      PLOT ( XADDR ( XORT ), Y
ADDR ( YORT ), 2 );
[2E9] 316:      CIRCLE ( XADDR ( XORT ),
YADDR ( YORT ), 2, 0 );
[B68] 317:      AKTPUNKT [ XORT, YORT ]
:= TRUE
[S72] 318:      END; ( IF )
[NO2] 319:      END; ( + )
[WW3] 320:      '-' : ZEICHNEPLUS;
[B21] 321:      BEGIN
[RR8] 322:      IF AKTPUNKT [ XORT, YORT ] THEN
BEGIN
[558] 323:      AKTPUNKT [ XORT, YORT ] := FA
LSE;
[BV5] 324:      CHUNVERAENDERT := FALSE;
[168] 325:      PUNKT ( XADDR ( XORT ), YADDR
( YORT ), 0 );
[023] 326:      BILDSCHIRMAUFBAU;
[S72] 327:      END; ( IF )
[NO2] 328:      END; ( - )
[MG2] 329:      '2', '4', '6', '8',
[023] 330:      '7', '9', '1', '3' : BEGIN
[KB7] 331:      IF AKTPUNKT [ XOR
T, YORT ] THEN
[HE9] 332:      CIRCLE ( XADDR
( XORT ), YADDR ( YORT ), 2, 3 )
[ARO] 333:      ELSE
[2E9] 334:      CIRCLE ( XADD
R ( XORT ), YADDR ( YORT ), 2, 0 );
[VC2] 335:      CASE EING OF
[UE4] 336:      '4' : XORT
:= XORT - 1;
[PE4] 337:      '6' : XORT
:= XORT + 1;
[8F4] 338:      '8' : YORT
:= YORT + 1;
[4F4] 339:      '2' : YORT
:= YORT - 1;
[DX5] 340:      '7' : BEGI
N XORT := XORT - 1;

```

Mit uns muß man rechnen

MARLOW 88

CPU 8088-1, 4.77/10 MHz
640 KB RAM (256 KB bestückt)
8 Slots, 360 KB Floppy 5.25"
Herkules komp. Grafikkarte
parallele Schnittstelle
deutsche Tastatur, MS-DOS

MARLOW 286

CPU 80286, 6/10 MHz
512 KB RAM, 1.2 MB Floppy 5.25"
Herkules komp. Grafikkarte
parallele Schnittstelle
Echtzeituhr, Datum
deutsche Multifunktionstastatur,
220 Watt Netzteil, MS-DOS

MARLOW 386 Tower

CPU 80386, 16/20 MHz
Towergehäuse, 1.2 MB RAM
1.2 MB Floppy 5.25"
60 MB Festplatte
Herkules komp. Grafikkarte
parallele und serielle Schnittstelle,
Gameport, Echtzeituhr, Datum,
deutsche Multifunktionstastatur mit
abgesetztem Cursorblock,
220 Watt Netzteil, MS-DOS



MARLOW

Computersysteme erhalten Sie bei:

M.Kempenich
Zetastraße 13 a
6220 Rüdeshelm 4
06726/9987

Eckerth&Dorth
Unterm Buchwald 3 a
6296 Mengerskirchen
06476/8672

Alle **MARLOW** Computer werden mit ausführlichem deutschem Handbuch geliefert.
12 Monate Garantie. Eigene Service Werkstatt. Technische Änderungen vorbehalten.

Fordern Sie unsere Gesamtpreisliste an.
Händleranfragen mit Gewerbenachweis.


```

[353] 341:      YORT := YORT + 1
[TM0] 342:      END;
[DX5] 343:      '9' : BEGI
N XORT := XORT + 1;
[353] 344:      YORT := YORT + 1
[TM0] 345:      END;
[1X5] 346:      '1' : BEGI
N XORT := XORT - 1;
[F53] 347:      YORT := YORT - 1
[TM0] 348:      END;
[1X5] 349:      '3' : BEGI
N XORT := XORT + 1;
[F53] 350:      YORT := YORT - 1
[TM0] 351:      END;
[LA3] 352:      END; { 2. CA
SE )
[5Y5] 353:      IF XORT = 0
THEN XORT := 1;
[786] 354:      IF XORT = 13
THEN XORT := 12;
[496] 355:      IF YORT = 25
THEN YORT := 24;
[S46] 356:      IF YORT = 0
THEN YORT := 1;
[CN7] 357:      IF AKTPUNKT [ XOR
T, YORT ] THEN
[2E9] 358:      CIRCLE ( XADDR
( XORT ), YADDR ( YORT ), 2, 0 )
[GU9] 359:      ELSE CIRCLE ( XAD
DR ( XORT ), YADDR ( YORT ), 2, 2 );
[TC9] 360:      IF ZEICHNE_PLUS T
HEN GOTO ZEICHNEPLUS;
[LV9] 361:      IF ZEICHNE_MINUS
THEN GOTO ZEICHNEMINUS;
[TM0] 362:      END;
[LA3] 363:      END; { 1. CASE }
[VA3] 364:      END; { WHILE }
[D7D] 365:      ENDE: MEM[0:$417] := NUMLOCK; { NUM LOCK IN
ALTEN ZUSTAND }
[2M5] 366:      IF NOT UNVERAENDERT THEN
[B21] 367:      BEGIN
[YD6] 368:      ASSIGN ( DATEI, 'CONST.DTA' );
[944] 369:      REWRITE ( DATEI );
[W64] 370:      FOR CLAUF := 1 TO 3 DO
[B21] 371:      BEGIN
[LR2] 372:      CASE CLAUF OF
[X33] 373:      1 : Writeln ( DATEI,
[MHB] 374:      'CONST OBENPUNKTE1 : ARRA
Y [ 1..255, 1..12 ] OF BYTE = ');
[AX3] 375:      2 : Writeln ( DATEI,
[MLA] 376:      'OBENPUNKTE2 : ARRA
Y [ 1..255, 1..12 ] OF BYTE = ');
[BX3] 377:      3 : Writeln ( DATEI,
[PPA] 378:      'UNTENPUNKTE : ARRA
Y [ 1..255, 1..12 ] OF BYTE = ');
[2X2] 379:      END; { CASE }
[ZK3] 380:      WRITE ( DATEI, '(' );
[UC4] 381:      FOR ALAUF := 1 TO 255 DO
[B21] 382:      BEGIN
[ND3] 383:      WRITE ( DATEI, '(' );
[CK4] 384:      FOR BLAUF := 1 TO 12 DO
[B21] 385:      BEGIN
[LR2] 386:      CASE CLAUF OF
[72B] 387:      1 : WRITE ( DATEI, OBENPUN
KTE_1 [ BLAUF, ALAUF ] );
[F2B] 388:      2 : WRITE ( DATEI, OBENPUN
KTE_2 [ BLAUF, ALAUF ] );
[RCB] 389:      3 : WRITE ( DATEI, UNTENPU
NKTE_ [ BLAUF, ALAUF ] );
[2X2] 390:      END; { CASE }
[7V7] 391:      IF BLAUF <> 12 THEN WRITE ( D
ATEI, ', ' );
[EB3] 392:      END; { BLAUF }
[383] 393:      WRITE ( DATEI, ' ' );
[U58] 394:      IF ALAUF <> 255 THEN Writeln ( DAT
EI, ', ' );
[EB3] 395:      END; { ALAUF }
[MM3] 396:      WRITE ( DATEI, ' ' );
[P6C] 397:      IF CLAUF <> 3 THEN FOR ALAUF := 1 TO 5
DO Writeln ( DATEI );
[EB3] 398:      END; { CLAUF }
[733] 399:      CLOSE ( DATEI );
[NL5] 400:      END; { IF UNVERAENDERT }
[D32] 401:      TEXTMODE
[SLO] 402:      END.

```

Listing 1. Der Quellcode des Zeichensatzeditors
»prn_def.pas«. Zur Compilierung aller drei Pascal-Program-
me benötigen Sie die Datei »const.dta« aus Listing 2.

```

[71B] 0: CONST OBENPUNKTE1 : ARRAY [ 1..255, 1..12 ]
OF BYTE =
[087] 1: ( ( 31, 32, 32, 32, 34, 34, 34, 32, 32, 32,
31, 0 ),
[VG7] 2: ( 31, 63, 37, 37, 36, 60, 36, 37, 37, 63, 3
1, 0 ),
[YD7] 3: ( 24, 28, 62, 62, 31, 15, 31, 62, 62, 28, 2
4, 0 ),
[CB7] 4: ( 0, 12, 12, 30, 30, 63, 30, 30, 12, 12, 0,
0 ),
[S87] 5: ( 0, 4, 12, 12, 52, 62, 62, 52, 12, 12, 4,
0 ),
[SA7] 6: ( 6, 14, 30, 62, 62, 127, 62, 62, 30, 14, 6
, 0 ),
[1D7] 7: ( 8, 28, 30, 30, 62, 62, 62, 30, 30, 28, 8,
0 ),
[YM5] 8: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 9: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 10: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 11: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 12: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 13: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 14: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 15: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[L57] 16: ( 0, 62, 30, 30, 28, 28, 12, 12, 8, 8, 0, 0
),
[YM5] 17: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 18: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 19: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 20: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[E47] 21: ( 0, 1, 8, 4, 112, 2, 145, 132, 40, 64, 0,
0 ),
[P57] 22: ( 0, 0, 30, 30, 30, 30, 30, 30, 30, 30, 0,
0 ),
[666] 23: ( 0, 0, 4, 64, 2, 254, 2, 64, 4, 0, 0, 0 ),
[YM5] 24: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 25: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[HT5] 26: ( 0, 0, 0, 0, 0, 0, 0, 0, 36, 0, 24, 0 ),
[YM5] 27: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[6R5] 28: ( 0, 0, 30, 2, 2, 2, 2, 2, 2, 2, 0, 0 ),
[W16] 29: ( 8, 8, 28, 8, 8, 8, 8, 8, 8, 28, 8, 8 ),
[Z96] 30: ( 0, 0, 4, 12, 28, 60, 28, 12, 4, 0, 0, 0 ),
[RD7] 31: ( 0, 32, 48, 56, 60, 60, 60, 56, 48, 32, 0,
0 ),
[YM5] 32: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[W46] 33: ( 0, 0, 0, 0, 0, 5, 88, 112, 0, 0, 0, 0 ),
[LK6] 34: ( 0, 0, 0, 64, 128, 0, 0, 64, 128, 0, 0, 0
),
[M27] 35: ( 0, 18, 18, 63, 0, 0, 0, 0, 127, 36, 36, 0
),
[OR6] 36: ( 0, 0, 1, 48, 0, 75, 92, 224, 69, 2, 0, 0
),
[8E7] 37: ( 64, 128, 129, 66, 4, 8, 16, 32, 65, 130,
2, 1 ),
[OC6] 38: ( 0, 2, 5, 0, 56, 9, 68, 82, 33, 5, 0, 0 ),
[666] 39: ( 0, 0, 0, 0, 160, 192, 0, 0, 0, 0, 0, 0 ),
[3E6] 40: ( 0, 0, 0, 0, 12, 50, 65, 0, 128, 0, 0, 0 ),
[S96] 41: ( 0, 0, 0, 0, 0, 129, 2, 76, 48, 0, 0, 0 ),
[HK6] 42: ( 0, 0, 40, 0, 16, 124, 16, 0, 40, 0, 0, 0
),
[HR5] 43: ( 0, 0, 0, 0, 0, 30, 0, 0, 0, 0, 0, 0 ),
[KN5] 44: ( 0, 0, 0, 0, 2, 3, 0, 0, 0, 0, 0, 0 ),
[YM5] 45: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[GN5] 46: ( 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0 ),
[7M6] 47: ( 0, 0, 1, 2, 4, 8, 16, 32, 64, 128, 0, 0 ),
[827] 48: ( 0, 28, 34, 1, 64, 64, 64, 1, 34, 28, 0, 0
),
[AG6] 49: ( 0, 0, 32, 32, 32, 127, 0, 0, 0, 0, 0, 0 ),
[3M6] 50: ( 0, 0, 33, 2, 64, 68, 64, 64, 8, 48, 0, 0
),
[1L6] 51: ( 0, 0, 34, 3, 64, 64, 72, 72, 1, 54, 0, 0
),
[OF6] 52: ( 0, 2, 2, 6, 10, 18, 34, 127, 2, 2, 0, 0 ),
[P27] 53: ( 0, 0, 64, 57, 40, 40, 40, 33, 70, 0, 0, 0
),
[HB6] 54: ( 0, 0, 62, 9, 72, 72, 72, 1, 38, 0, 0, 0 ),
[7E7] 55: ( 0, 0, 96, 72, 72, 79, 72, 88, 104, 72, 0,
0 ),
[UA6] 56: ( 0, 0, 54, 1, 72, 72, 72, 1, 54, 0, 0, 0 ),
[G17] 57: ( 0, 0, 50, 11, 64, 64, 65, 10, 60, 0, 0, 0
),
[CV5] 58: ( 0, 0, 0, 0, 2, 18, 16, 0, 0, 0, 0, 0 ),
[ET5] 59: ( 0, 0, 0, 0, 2, 11, 8, 0, 0, 0, 0, 0 ),

```


[826] 60: (0, 0, 0, 24, 0, 36, 0, 66, 0, 0, 0, 0),
 [PV5] 61: (0, 0, 0, 8, 8, 8, 8, 8, 8, 0, 0),
 [M26] 62: (0, 0, 0, 66, 0, 36, 0, 24, 0, 0, 0, 0),
 [F47] 63: (0, 0, 0, 65, 1, 137, 128, 144, 0, 96, 0, 0, 0),
 [6R7] 64: (60, 66, 0, 153, 133, 165, 185, 5, 68, 56, 0, 0),
 [8F6] 65: (0, 5, 6, 4, 12, 20, 36, 4, 126, 1, 0, 0, 0)
 [CM6] 66: (0, 0, 7, 25, 33, 64, 72, 72, 0, 53, 2, 0),
 [DN6] 67: (0, 0, 14, 17, 32, 0, 64, 65, 0, 32, 0, 0),
 [3X6] 68: (0, 0, 7, 25, 33, 64, 64, 64, 1, 34, 28, 0),
 [1R6] 69: (0, 2, 21, 32, 8, 64, 64, 65, 0, 32, 0, 0),
 [AG7] 70: (0, 1, 78, 120, 72, 72, 72, 64, 64, 64, 64, 0),
 [MP6] 71: (0, 0, 25, 36, 0, 64, 69, 66, 12, 0, 0, 0),
 [B67] 72: (0, 9, 14, 24, 104, 8, 8, 9, 14, 24, 96, 0),
 [NR6] 73: (0, 0, 33, 0, 0, 64, 65, 70, 88, 96, 0, 0),
 [NU6] 74: (0, 0, 0, 1, 32, 0, 2, 67, 68, 92, 100, 0),
 [YN6] 75: (0, 0, 7, 28, 96, 0, 12, 18, 33, 64, 0, 0),
 [8S5] 76: (0, 0, 7, 28, 96, 0, 0, 1, 1, 0, 0, 0),
 [1A7] 77: (3, 28, 32, 64, 32, 24, 0, 16, 1, 38, 24, 0),
 [KT6] 78: (0, 3, 4, 24, 32, 0, 127, 4, 8, 16, 96, 0),
 [G77] 79: (0, 14, 17, 32, 0, 64, 65, 66, 4, 56, 0, 0),
 [BE7] 80: (0, 32, 16, 67, 92, 72, 72, 72, 0, 48, 0, 0),
 [587] 81: (0, 14, 19, 32, 0, 64, 67, 67, 4, 57, 0, 0),
 [9C7] 82: (0, 32, 16, 67, 76, 80, 64, 76, 10, 49, 0, 0),
 [OH6] 83: (0, 0, 1, 48, 0, 72, 72, 64, 69, 2, 0, 0, 0),
 [PV6] 84: (0, 0, 0, 3, 12, 16, 0, 64, 64, 64, 64, 0),
 [8B6] 85: (0, 6, 25, 96, 0, 0, 1, 2, 31, 96, 0, 0),
 [CT6] 86: (0, 0, 112, 12, 3, 1, 2, 4, 24, 32, 64, 0),
 [RT6] 87: (0, 62, 65, 1, 2, 28, 33, 1, 2, 4, 120, 0),
 [H27] 88: (0, 1, 66, 32, 20, 12, 8, 2, 17, 32, 64, 0),
 [ST6] 89: (0, 0, 24, 100, 0, 0, 0, 5, 14, 48, 64, 0),
 [1P6] 90: (0, 1, 2, 5, 1, 73, 64, 80, 96, 64, 0, 0),
 [N67] 91: (0, 0, 0, 255, 128, 128, 128, 128, 0, 0, 0, 0),
 [3K6] 92: (0, 0, 128, 64, 32, 16, 8, 4, 2, 1, 0, 0, 0),
 [H67] 93: (0, 0, 0, 128, 128, 128, 128, 255, 0, 0, 0, 0),
 [5W6] 94: (0, 32, 0, 64, 0, 128, 0, 64, 0, 32, 0, 0),
 [YM5] 95: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
 [D56] 96: (0, 0, 0, 0, 192, 160, 0, 0, 0, 0, 0, 0, 0),
 [TS5] 97: (0, 0, 2, 5, 0, 8, 8, 9, 3, 4, 8, 1),
 [B25] 98: (0, 0, 6, 25, 100, 0, 0, 1, 6, 0, 0, 0),
 [UR5] 99: (0, 0, 2, 5, 0, 8, 8, 8, 1, 1, 1, 1),
 [2W5] 100: (0, 0, 2, 5, 0, 8, 1, 3, 28, 96, 0, 1),
 [WT5] 101: (0, 0, 0, 6, 1, 8, 10, 4, 0, 0, 0, 1),
 [VX6] 102: (0, 0, 0, 3, 12, 56, 72, 66, 16, 32, 0),
 [286] 103: (0, 0, 0, 4, 3, 11, 11, 11, 12, 8, 0, 2),
 [R76] 104: (0, 0, 0, 3, 12, 56, 64, 8, 11, 4, 0, 1),
 [246] 105: (0, 0, 1, 0, 2, 36, 41, 32, 0, 0, 0, 1),
 [RY5] 106: (0, 0, 0, 1, 1, 5, 5, 38, 40, 32, 0, 2),
 [086] 107: (0, 0, 0, 3, 12, 56, 66, 10, 9, 4, 0, 0),
 [P56] 108: (0, 0, 1, 0, 2, 7, 88, 112, 0, 0, 0, 1),
 [WZ5] 109: (0, 3, 4, 8, 8, 11, 4, 8, 8, 11, 4, 1),
 [9U5] 110: (0, 0, 0, 11, 4, 8, 8, 8, 3, 4, 0, 1),
 [UR5] 111: (0, 0, 2, 5, 0, 0, 8, 9, 2, 4, 0, 0),
 [1T5] 112: (0, 0, 0, 11, 14, 0, 8, 2, 4, 0, 0, 0),
 [V36] 113: (0, 0, 0, 4, 2, 10, 10, 11, 4, 8, 0, 1),
 [XV5] 114: (0, 0, 3, 4, 8, 0, 11, 12, 0, 0, 0, 1),
 [UP5] 115: (1, 1, 0, 2, 0, 4, 1, 8, 4, 3, 0, 0),
 [B47] 116: (1, 1, 1, 16, 18, 22, 25, 112, 16, 16, 16, 1)

[6T5] 117: (0, 0, 2, 13, 0, 0, 1, 3, 12, 0, 0, 1),
 [9S5] 118: (0, 0, 8, 4, 3, 1, 0, 2, 4, 8, 0, 0),
 [XU5] 119: (0, 3, 12, 0, 1, 2, 5, 0, 1, 2, 12, 0),
 [UR5] 120: (0, 0, 8, 1, 4, 2, 0, 5, 0, 8, 0, 0),
 [3T5] 121: (0, 0, 0, 0, 7, 8, 0, 3, 4, 8, 0, 2),
 [366] 122: (0, 0, 0, 0, 9, 11, 11, 13, 9, 8, 0, 0),
 [DB6] 123: (0, 0, 0, 0, 0, 8, 119, 0, 128, 0, 0, 0),
 [CT5] 124: (0, 0, 0, 0, 0, 247, 0, 0, 0, 0, 0, 0),
 [ZA6] 125: (0, 0, 0, 128, 0, 119, 8, 0, 0, 0, 0, 0),
 [K57] 126: (0, 64, 0, 128, 128, 0, 64, 0, 0, 64, 0, 0),
 [RY6] 127: (51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34, 0),
 [DN6] 128: (0, 0, 14, 17, 32, 0, 64, 65, 0, 32, 0, 0),
 [3T5] 129: (0, 0, 2, 13, 0, 0, 1, 3, 12, 0, 1, 0),
 [RU5] 130: (0, 0, 0, 6, 1, 8, 42, 4, 0, 0, 0, 1),
 [9W5] 131: (0, 0, 2, 5, 0, 40, 8, 41, 3, 4, 8, 1),
 [PS5] 132: (0, 0, 2, 5, 0, 8, 8, 9, 3, 4, 9, 0),
 [ZV5] 133: (0, 0, 2, 5, 0, 40, 8, 9, 3, 4, 8, 1),
 [OB6] 134: (0, 0, 2, 5, 0, 40, 24, 25, 35, 4, 8, 1),
 [UR5] 135: (0, 0, 2, 5, 0, 8, 8, 8, 1, 1, 1, 1),
 [V56] 136: (0, 0, 0, 6, 17, 8, 42, 4, 16, 0, 0, 1),
 [PV5] 137: (2, 2, 2, 6, 3, 10, 10, 4, 1, 1, 0, 2),
 [FU5] 138: (0, 0, 0, 6, 1, 40, 10, 4, 0, 0, 0, 1),
 [WV5] 139: (0, 0, 1, 0, 34, 4, 9, 0, 32, 0, 0, 1),
 [946] 140: (0, 0, 4, 4, 0, 40, 15, 32, 0, 1, 0, 2),
 [DU5] 141: (0, 0, 1, 0, 2, 36, 9, 0, 0, 0, 0, 1),
 [FZ6] 142: (0, 5, 6, 4, 12, 20, 36, 132, 126, 1, 128, 0),
 [S67] 143: (0, 5, 6, 4, 12, 20, 100, 132, 190, 65, 0, 0),
 [4G7] 144: (0, 2, 21, 32, 8, 64, 192, 193, 128, 160, 0, 0),
 [E76] 145: (0, 5, 10, 10, 9, 6, 7, 2, 10, 8, 5, 2),
 [B07] 146: (5, 6, 4, 12, 20, 36, 4, 126, 69, 68, 68, 0),
 [956] 147: (0, 0, 2, 5, 16, 0, 40, 9, 18, 4, 0, 0),
 [UR5] 148: (0, 0, 2, 5, 0, 0, 8, 9, 2, 4, 0, 0),
 [AV5] 149: (0, 0, 2, 5, 0, 32, 8, 9, 2, 4, 0, 0),
 [W56] 150: (0, 0, 2, 13, 0, 32, 1, 35, 12, 0, 0, 1),
 [C26] 151: (0, 0, 2, 13, 0, 32, 1, 3, 12, 0, 0, 1),
 [3T5] 152: (0, 0, 0, 0, 7, 8, 0, 3, 4, 8, 0, 2),
 [HF7] 153: (0, 14, 17, 32, 0, 192, 65, 66, 132, 56, 0, 0),
 [Z57] 154: (0, 6, 25, 96, 0, 128, 1, 2, 159, 96, 0, 0),
 [1V5] 155: (0, 0, 2, 5, 0, 11, 12, 8, 1, 1, 1, 1),
 [496] 156: (0, 0, 4, 5, 14, 52, 4, 64, 64, 0, 0, 0),
 [RY6] 157: (51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34, 0),
 [RY6] 158: (51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34, 0),
 [RY6] 159: (51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34, 0),
 [RY6] 160: (51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34, 0),
 [RY6] 161: (51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34, 0),
 [RY6] 162: (51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34, 0),
 [RY6] 163: (51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34, 0),
 [RY6] 164: (51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34, 0),
 [RY6] 165: (51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34, 0),
 [RY6] 166: (51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34, 0),
 [RY6] 167: (51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34, 0),
 [RY6] 168: (51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34, 0),
 [8P5] 169: (0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
 [5N5] 170: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7),
 [F86] 171: (0, 2, 112, 4, 0, 8, 0, 16, 2, 40, 4, 0),
 [DF6] 172: (0, 2, 112, 4, 0, 10, 2, 22, 2, 47, 2, 0),
 [CS5] 173: (0, 0, 0, 0, 0, 95, 0, 0, 0, 0, 0, 0),
 [8H6] 174: (0, 0, 24, 0, 36, 0, 90, 0, 36, 0, 66, 0),
 [DL6] 175: (0, 66, 0, 36, 0, 90, 0, 36, 0, 24, 0, 0),
 [P66] 176: (0, 36, 0, 0, 36, 0, 0, 36, 0, 0, 36, 0),
 [UM8] 177: (146, 36, 73, 146, 36, 73, 146, 36, 73, 14, 6, 36, 73)


```

[A57] 178: ( 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255,
0 ),
[7T5] 179: ( 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0 ),
[AW5] 180: ( 1, 1, 1, 1, 1, 1, 255, 0, 0, 0, 0, 0 ),
[GW5] 181: ( 5, 5, 5, 5, 5, 255, 0, 0, 0, 0, 0, 0 ),
[N36] 182: ( 1, 1, 1, 1, 1, 1, 255, 0, 0, 0, 255, 0, 0 ),

[3P5] 183: ( 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0 ),
[RY6] 184: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[166] 185: ( 5, 5, 5, 5, 5, 253, 0, 0, 0, 255, 0, 0 ),

[RY6] 186: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 187: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 188: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 189: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 190: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 191: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 192: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 193: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 194: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 195: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 196: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 197: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 198: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 199: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 200: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 201: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 202: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 203: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 204: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 205: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 206: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 207: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 208: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 209: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 210: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 211: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 212: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 213: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 214: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 215: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 216: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 217: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 218: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[K1A] 219: ( 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
55, 255, 255, 255 ),
[RY6] 220: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 221: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 222: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 223: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 224: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[UP6] 225: ( 0, 3, 12, 49, 0, 64, 65, 14, 48, 0, 0, 0 ),
[RY6] 226: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),

```

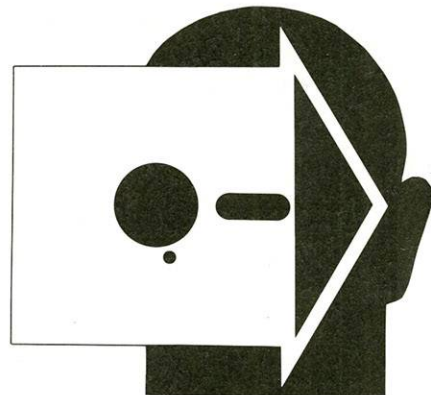
```

[RY6] 227: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 228: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 229: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 230: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 231: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 232: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 233: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 234: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 235: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[RY6] 236: ( 51, 0, 68, 1, 62, 8, 60, 10, 65, 81, 34,
0 ),
[OD7] 237: ( 28, 34, 0, 65, 97, 81, 81, 16, 34, 60, 2,
0 ),
[677] 238: ( 33, 18, 64, 65, 94, 64, 9, 41, 17, 0, 2,
0 ),
[WC7] 239: ( 28, 34, 0, 65, 65, 69, 65, 2, 34, 28, 2,
0 ),
[MY6] 240: ( 0, 64, 64, 79, 72, 8, 0, 55, 0, 1, 2, 32 ),
[SP6] 241: ( 18, 1, 0, 0, 63, 0, 65, 65, 80, 32, 2, 0 ),
[H17] 242: ( 17, 2, 64, 1, 14, 32, 33, 1, 1, 32, 2, 32 ),
[FD6] 243: ( 64, 62, 1, 0, 0, 0, 1, 126, 0, 0, 2, 64 ),

[076] 244: ( 0, 0, 0, 3, 4, 56, 64, 8, 11, 4, 0, 0 ),
[A07] 245: ( 126, 1, 0, 0, 15, 0, 0, 1, 127, 0, 66, 64 ),
[T47] 246: ( 0, 68, 1, 62, 8, 60, 10, 65, 81, 34, 0, 3
4 ),
[YM5] 247: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[R97] 248: ( 98, 68, 64, 66, 70, 72, 80, 96, 66, 0, 4,
0 ),
[C56] 249: ( 0, 0, 0, 0, 0, 0, 255, 0, 0, 255, 0, 1 ),
[LB6] 250: ( 128, 64, 32, 16, 8, 4, 2, 1, 0, 0, 0, 0 ),
[T67] 251: ( 0, 0, 128, 128, 128, 128, 255, 0, 0, 0, 0,
0 ),
[3Y6] 252: ( 32, 0, 64, 0, 128, 0, 64, 0, 32, 0, 0, 32 ),
[YM5] 253: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 254: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[BV5] 255: ( 4, 6, 1, 8, 8, 8, 1, 14, 0, 0, 2, 4 );
[000] 256:
[000] 257:
[000] 258:
[000] 259:
[ZL9] 260: OBNPUNKTE2 : ARRAY [ 1..255, 1..12 ]
OF BYTE =
[FD6] 261: ( 62, 1, 1, 21, 1, 1, 1, 21, 1, 1, 62, 0 )
[7H7] 262: ( 62, 63, 43, 41, 45, 61, 45, 41, 43, 63, 6
2, 0 ),
[EC7] 263: ( 24, 60, 60, 62, 62, 31, 62, 62, 60, 60, 2
4, 0 ),
[787] 264: ( 8, 8, 28, 28, 62, 62, 62, 28, 28, 8, 8, 0
),
[VC7] 265: ( 0, 8, 12, 44, 40, 126, 126, 40, 44, 12, 8
0 ),
[HP7] 266: ( 4, 30, 62, 60, 124, 127, 124, 60, 62, 30,
4, 0 ),
[HE7] 267: ( 8, 28, 60, 60, 62, 62, 62, 60, 60, 28, 8,
0 ),
[YM5] 268: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 269: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 270: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 271: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 272: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 273: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 274: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 275: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[777] 276: ( 0, 62, 60, 60, 28, 28, 24, 24, 8, 8, 0, 0 ),
[YM5] 277: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 278: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 279: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 280: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[2C7] 281: ( 0, 1, 10, 16, 68, 160, 7, 16, 136, 64, 0,
0 ),
[Y77] 282: ( 0, 0, 60, 60, 60, 60, 60, 60, 60, 60, 0,
0 ),
[W67] 283: ( 0, 1, 65, 5, 129, 255, 129, 5, 65, 1, 0,
0 ),

```


Sesam öffne dich!



BCi DisAsm erschließt jedes Programm!

Wandelt dies
in diese Form um!

```
D>DEBUG HALLO.COM
-D100 160
1C80:0100 EB 43 90 48 61 6C 6C 6F-20 46 72 65 75 6E 64 65 .C.Hallo Freunde
1C80:0110 21 24 0D 0A 56 69 65 6C-20 45 72 66 6F 6C 67 20 !$.Viel Erfolg
1C80:0120 6D 69 74 20 42 43 49 20-44 69 73 41 73 6D 2F 69 mit BCI DisAsm/i
1C80:0130 38 36 2E 24 0D 0A 47 72-81 E1 65 20 76 6F 6E 20 86$.Gr...e von
1C80:0140 42 43 49 2E 24 BA 03 01-B4 09 CD 21 BA 12 01 B4 BCI$......!
1C80:0150 09 CD 21 BA 34 01 B4 09-CD 21 CD 20 96 B9 B9 10 ...!.4.....!
1C80:0160 00
```

und kann nach Modifizierung oder Änderung durch einen Editor mit gängigen Assemblern oder Debuggern wieder assembliert werden.

BCi DisAsm

Es gibt kaum einen, der ihm das Wasser reichen kann!

Auch Experten kennen nicht immer die Lösung, aber sie wissen, wo sie suchen müssen.

BCi DisAsm hilft JEDEM, die verwirrenden und frustrierenden Software-Puzzle, welche sich beim Umgang mit dem Computer ergeben, zu lösen. Und zwar schneller als mit jedem anderen Produkt, das Sie erwerben können, egal welche Preisklasse Sie betrachten.

Mit dem BCI DisAsm erwerben Sie Know-How, das Sie sonst Jahre an Erfahrung kosten würde. Versuchen Sie nicht, das Rad neu zu erfinden!

BCi DisAsm – Der intelligente Disassembler

BCi DisAsm ist ein intelligenter, selbsttätig 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 Menues gesteuert werden. Die Menuestruktur ist Lotus-kompatibel, selbstverständlich in Deutsch. Es besteht keine Notwendigkeit, neue Befehle zu lernen oder ständig im Handbuch nachzuschlagen. BCI DisAsm ist unmittelbar anwendbar!

Minimale Systemanforderungen

256 kB RAM, 8088/8086/80186/80286 oder 80386

CPU, MS-DOS oder PC-DOS Version 2.0 oder größer

```
BCI DisAsm/i86 CROSSREFERENZ Version 1.0 - HALLO.XRF Seite 1

00103h : 145
00112h : 14C
00134h : 153
BIOS-TERM_norm:20h : 15A
DOS1-Dsp_Str : 14A 151 158
H00145 : 100

ANZAHL BEGRIFFE AUF DER CROSSREFERENZLISTE: 6 - 4

;NAME: HALLO.COM
;DATEILAENGE: 0005Ch - 92d
;START (CS:IP): 00100h
;CODE ENDE: 0015Ch
;CODE ANFANG: 00100h
;DATUM: Tue Dec 01 07:14:28 1987

CODE SEGMENT BYTE PUBLIC 'CODE'
ASSUME CS:CODE,DS:CODE,ES:NOTHING,SS:NOTHING

P00100 PROC
ORG 0100h

H00100: JMP Short H00145 ;00100 EB43 _C
;NOP ;00102 90 _
;DB "Hallo Freunde!$" ;00103 48616C6C6F2046
;DB 0Dh ;00112
;DB 0Ah ;00113
;DB "Viel Erfolg mit BCI DisAsm/i86.$" ;00114 5669656C204572
;DB 0Dh ;00134
;DB 0Ah ;00135
;DB "Gr" ;00136 4772
;DB 81h ;00138
;DB 0E1h ;00139
;DB "e von BCI.$" ;0013A 6520766F6E2042

H00145: MOV DX,0103h ;00145 BA0301
MOV AH,09h ;00148 B409
INT 21h ;0014A CD21
MOV DX,0112h ;1-Dsp_Str ;0014C BA1201
MOV AH,09h ;0014F B409
INT 21h ;00151 CD21
MOV DX,0134h ;1-Dsp_Str ;00153 BA3401
MOV AH,09h ;00156 B409
INT 21h ;1-Dsp_Str ;00158 CD21
INT 20h ;B-TERM_norm:20h ;0015A CD20

P00100 ENDP
CODE ENDS
END H00100
```

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 etwas 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 compilierten 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.

BCi DisAsm kostet incl. einem sehr ausführlichen deutschen Handbuch (mehr als 100 Seiten) nur

(Best.-Nr. 227)

199,- DM*
*unverbindliche Preisempfehlung

Bestellen Sie noch heute mittels der entsprechenden Postkarte bei:

DMV
Software

Postfach 250, 3440 Eschwege, Tel. 05651/8702
Händleranfragen willkommen


```

[YM5] 284: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 285: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[AK7] 286: ( 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 56, 1
        6, 16 ),
[YM5] 287: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[NN5] 288: ( 0, 0, 60, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[CU5] 289: ( 0, 24, 0, 0, 0, 0, 0, 0, 0, 0, 24, 0 ),
[N47] 290: ( 0, 4, 12, 28, 60, 60, 60, 28, 12, 4, 0, 0 ),
[G47] 291: ( 0, 0, 32, 56, 56, 60, 56, 48, 32, 0, 0, 0 ),
[YM5] 292: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[F56] 293: ( 0, 0, 0, 0, 1, 13, 48, 96, 0, 0, 0, 0 ),
[LK6] 294: ( 0, 0, 0, 64, 128, 0, 0, 64, 128, 0, 0, 0 ),
[1A7] 295: ( 0, 0, 0, 127, 36, 36, 36, 36, 126, 0, 0,
        0 ),
[SL6] 296: ( 0, 0, 0, 33, 81, 3, 29, 233, 0, 70, 0, 0 ),
[W97] 297: ( 128, 64, 65, 130, 4, 8, 16, 32, 66, 129,
        1, 2 ),
[AH6] 298: ( 0, 6, 0, 9, 33, 88, 22, 2, 100, 0, 2, 0 ),
[SD6] 299: ( 0, 0, 0, 32, 128, 192, 0, 0, 0, 0, 0, 0 ),
[ZD6] 300: ( 0, 0, 0, 0, 12, 50, 64, 129, 0, 0, 0, 0 ),
[4A6] 301: ( 0, 0, 0, 0, 1, 0, 130, 76, 48, 0, 0, 0 ),
[BD7] 302: ( 0, 72, 0, 48, 16, 120, 16, 48, 0, 72, 0,
        0 ),
[XO6] 303: ( 0, 8, 8, 8, 8, 62, 8, 8, 8, 8, 0, 0 ),
[KN5] 304: ( 0, 0, 0, 0, 2, 3, 0, 0, 0, 0, 0, 0 ),
[YV5] 305: ( 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 0, 0 ),
[GN5] 306: ( 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0 ),
[7M6] 307: ( 0, 0, 1, 2, 4, 8, 16, 32, 64, 128, 0, 0 ),
[EU6] 308: ( 0, 28, 34, 64, 1, 1, 1, 64, 34, 28, 0, 0 ),
[Z16] 309: ( 0, 0, 1, 1, 65, 127, 1, 1, 1, 0, 0, 0 ),
[Z66] 310: ( 0, 0, 0, 35, 97, 5, 1, 9, 9, 65, 49, 0, 0 ),
[7C6] 311: ( 0, 0, 34, 96, 1, 1, 1, 1, 88, 38, 0, 0 ),
[2F6] 312: ( 0, 0, 4, 8, 16, 33, 65, 127, 1, 1, 0, 0 ),
[H76] 313: ( 0, 0, 74, 122, 1, 1, 1, 8, 70, 0, 0, 0 ),
[BG6] 314: ( 0, 0, 62, 72, 1, 1, 1, 104, 38, 0, 0, 0 ),
[UC6] 315: ( 0, 0, 64, 0, 0, 15, 16, 32, 64, 0, 0, 0 ),
[LC6] 316: ( 0, 0, 38, 88, 1, 1, 1, 88, 38, 0, 0, 0 ),
[CC6] 317: ( 0, 0, 50, 64, 9, 9, 8, 66, 60, 0, 0, 0 ),
[CV5] 318: ( 0, 0, 0, 0, 2, 18, 16, 0, 0, 0, 0, 0 ),
[ET5] 319: ( 0, 0, 0, 0, 2, 11, 8, 0, 0, 0, 0, 0 ),
[5C6] 320: ( 0, 0, 16, 0, 40, 0, 68, 0, 130, 0, 0, 0 ),
[PR5] 321: ( 0, 0, 4, 4, 4, 4, 4, 4, 4, 0, 0, 0 ),
[ZA6] 322: ( 0, 0, 130, 0, 68, 0, 40, 0, 16, 0, 0, 0 ),
[KN6] 323: ( 0, 0, 0, 131, 8, 16, 0, 160, 64, 0, 0, 0 ),
[1A7] 324: ( 56, 68, 130, 24, 32, 0, 56, 128, 66, 56,
        0, 0 ),
[KK6] 325: ( 0, 1, 6, 0, 8, 16, 32, 64, 126, 1, 0, 0 ),
[796] 326: ( 0, 0, 7, 56, 64, 1, 1, 1, 89, 32, 6, 0 ),
[XK6] 327: ( 0, 0, 14, 16, 33, 65, 1, 0, 66, 0, 0, 0 ),
[GK6] 328: ( 0, 0, 7, 56, 64, 1, 1, 1, 64, 34, 28, 0 ),
[9A6] 329: ( 0, 6, 32, 25, 73, 9, 1, 0, 66, 0, 0, 0 ),
[BS5] 330: ( 0, 3, 4, 56, 0, 0, 0, 0, 0, 0, 0, 0 ),
[3C6] 331: ( 0, 0, 24, 33, 69, 5, 0, 14, 72, 0, 0, 0 ),
[U46] 332: ( 0, 3, 4, 56, 64, 0, 0, 3, 4, 56, 64, 0 ),
[N76] 333: ( 0, 0, 0, 65, 65, 1, 2, 4, 56, 64, 0, 0 ),
[CL6] 334: ( 0, 0, 0, 1, 34, 66, 64, 7, 4, 56, 64, 0 ),
[ZH6] 335: ( 0, 0, 7, 56, 72, 8, 4, 18, 33, 64, 0, 0 ),
[KS5] 336: ( 0, 0, 7, 57, 65, 1, 1, 0, 0, 0, 0, 0 ),
[X57] 337: ( 3, 28, 32, 64, 32, 8, 16, 0, 33, 6, 120,
        0 ),

```

```

[ZX6] 338: ( 0, 3, 4, 24, 32, 64, 127, 4, 8, 16, 96, 0
        ),
[FV6] 339: ( 0, 14, 16, 33, 65, 1, 0, 2, 68, 56, 0, 0
        ),
[OV6] 340: ( 0, 32, 64, 3, 28, 32, 0, 0, 80, 32, 0, 0
        ),
[XX6] 341: ( 0, 14, 16, 37, 69, 5, 0, 2, 69, 56, 2, 0
        ),
[8X6] 342: ( 0, 32, 64, 3, 12, 48, 8, 4, 66, 48, 1, 0
        ),
[UX5] 343: ( 0, 0, 0, 33, 81, 1, 1, 9, 0, 70, 0, 0 ),
[XZ6] 344: ( 0, 64, 64, 67, 92, 96, 64, 0, 0, 0, 0, 0
        ),
[6B6] 345: ( 0, 6, 24, 97, 1, 1, 0, 2, 30, 97, 0, 0 ),
[9Y6] 346: ( 0, 0, 112, 12, 3, 0, 2, 12, 16, 32, 64, 0
        ),
[HS6] 347: ( 0, 62, 65, 0, 6, 30, 97, 0, 2, 4, 120, 0
        ),
[AV6] 348: ( 0, 1, 66, 36, 16, 8, 4, 18, 0, 33, 64, 0
        ),
[2M6] 349: ( 0, 0, 24, 96, 4, 4, 4, 1, 14, 112, 0, 0 ),
[DR6] 350: ( 0, 3, 69, 64, 72, 0, 17, 1, 33, 65, 0, 0
        ),
[XV5] 351: ( 0, 0, 0, 255, 1, 1, 1, 1, 0, 0, 0, 0 ),
[3K6] 352: ( 0, 0, 128, 64, 32, 16, 8, 4, 2, 1, 0, 0 ),
[HP5] 353: ( 0, 0, 0, 1, 1, 1, 1, 255, 0, 0, 0, 0 ),
[H17] 354: ( 32, 0, 64, 0, 128, 0, 128, 0, 64, 0, 32,
        0 ),
[YM5] 355: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[HD6] 356: ( 0, 0, 0, 0, 192, 128, 32, 0, 0, 0, 0, 0 ),
[HU5] 357: ( 0, 0, 2, 4, 9, 1, 1, 0, 10, 13, 1, 0 ),
[YV5] 358: ( 0, 0, 6, 24, 97, 9, 9, 8, 6, 0, 0, 0 ),
[ES5] 359: ( 0, 0, 2, 4, 9, 1, 1, 1, 8, 0, 0, 0 ),
[LY5] 360: ( 0, 0, 2, 4, 9, 1, 8, 2, 29, 97, 1, 0 ),
[YR5] 361: ( 0, 0, 0, 6, 8, 3, 1, 13, 1, 1, 1, 0 ),
[P77] 362: ( 0, 0, 10, 10, 11, 14, 114, 2, 16, 64, 32,
        0 ),
[3T5] 363: ( 0, 0, 0, 5, 8, 0, 0, 3, 6, 10, 2, 0 ),
[Z76] 364: ( 0, 0, 0, 3, 12, 48, 72, 0, 2, 13, 1, 0 ),
[DV5] 365: ( 0, 1, 0, 2, 0, 6, 104, 1, 1, 1, 1, 0 ),
[N26] 366: ( 0, 0, 1, 4, 4, 0, 1, 14, 106, 2, 2, 0 ),
[946] 367: ( 0, 0, 0, 3, 12, 48, 72, 2, 5, 8, 0, 0 ),
[E66] 368: ( 0, 1, 0, 2, 0, 14, 49, 97, 1, 1, 1, 0 ),
[L66] 369: ( 0, 11, 12, 0, 0, 3, 12, 0, 0, 2, 13, 0 ),
[2T5] 370: ( 0, 0, 0, 3, 12, 0, 0, 0, 10, 5, 1, 0 ),
[371] 371: ( 0, 0, 2, 4, 9, 1, 0, 10, 4, 0, 0, 0 ),
[MT5] 372: ( 0, 0, 0, 3, 12, 10, 2, 8, 4, 0, 0, 0 ),
[6U5] 373: ( 0, 0, 0, 4, 8, 1, 1, 3, 13, 9, 1, 0 ),
[GS5] 374: ( 1, 1, 2, 4, 8, 8, 2, 13, 1, 1, 1, 0 ),
[ZR5] 375: ( 0, 0, 2, 0, 4, 0, 8, 9, 5, 2, 0, 0 ),
[HV5] 376: ( 0, 0, 0, 2, 0, 6, 24, 97, 1, 1, 1, 0 ),
[ZS5] 377: ( 0, 0, 2, 12, 1, 1, 0, 2, 13, 1, 1, 0 ),
[HS5] 378: ( 0, 0, 8, 4, 3, 0, 2, 0, 4, 8, 0, 0 ),
[825] 379: ( 0, 2, 13, 1, 0, 2, 12, 1, 0, 2, 12, 0 ),
[ES5] 380: ( 0, 0, 1, 8, 2, 4, 6, 0, 9, 0, 0, 0 ),
[6W5] 381: ( 0, 0, 0, 1, 4, 10, 2, 3, 6, 10, 2, 0 ),
[WP5] 382: ( 0, 0, 0, 1, 0, 0, 4, 3, 8, 2, 2, 0 ),
[496] 383: ( 0, 0, 0, 0, 0, 0, 8, 119, 128, 0, 0, 0 ),
[CT5] 384: ( 0, 0, 0, 0, 0, 247, 0, 0, 0, 0, 0, 0 ),
[S66] 385: ( 0, 0, 0, 0, 128, 119, 8, 0, 0, 0, 0, 0 ),
[C37] 386: ( 0, 0, 128, 0, 0, 128, 0, 64, 64, 0, 128,
        0 ),
[N67] 387: ( 34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98,
        4 ),
[XK6] 388: ( 0, 0, 14, 16, 33, 65, 1, 0, 66, 0, 0, 0 ),
[S46] 389: ( 0, 0, 2, 12, 33, 1, 0, 34, 13, 1, 0, 0 ),
[746] 390: ( 0, 0, 0, 6, 40, 35, 1, 77, 1, 1, 1, 0 ),
[U96] 391: ( 0, 0, 2, 4, 41, 1, 65, 0, 42, 13, 1, 0 ),
[826] 392: ( 0, 0, 2, 4, 9, 33, 1, 0, 10, 45, 0, 0 ),
[VH6] 393: ( 0, 0, 2, 4, 73, 1, 33, 32, 10, 13, 1, 0 ),
[NN6] 394: ( 0, 0, 2, 4, 9, 33, 65, 64, 42, 13, 1, 0 ),
[ES5] 395: ( 0, 0, 2, 4, 9, 1, 1, 1, 8, 0, 0, 0 ),
[9W5] 396: ( 0, 0, 0, 6, 8, 35, 1, 45, 1, 1, 1, 0 ),
[G36] 397: ( 0, 0, 0, 38, 8, 1, 1, 13, 32, 0, 2, 0 ),
[C56] 398: ( 0, 0, 0, 6, 72, 3, 33, 45, 1, 1, 1, 0 ),
[CW5] 399: ( 0, 1, 0, 2, 32, 6, 8, 1, 33, 1, 1, 0 ),
[P76] 400: ( 4, 4, 0, 0, 40, 0, 78, 1, 33, 0, 2, 0 ),
[H46] 401: ( 0, 1, 0, 2, 64, 6, 40, 33, 1, 1, 1, 0 ),
[KK6] 402: ( 0, 1, 6, 0, 8, 16, 32, 64, 126, 1, 0, 0 )

```


[187] 403: (0, 1, 6, 0, 8, 16, 160, 64, 126, 129, 0, 0),
 [9A6] 404: (0, 6, 32, 25, 73, 9, 1, 0, 66, 0, 0, 0),
 [1V5] 405: (4, 10, 1, 1, 4, 14, 6, 9, 1, 5, 8, 2),
 [2G6] 406: (1, 6, 0, 8, 16, 32, 64, 126, 0, 1, 1, 0)
 [356] 407: (0, 0, 2, 4, 9, 41, 1, 32, 10, 4, 0, 0),
 [526] 408: (0, 0, 2, 4, 9, 41, 1, 0, 10, 36, 0, 0),
 [D96] 409: (0, 0, 2, 4, 73, 9, 33, 32, 10, 4, 0, 0),
 [766] 410: (0, 0, 2, 12, 33, 1, 64, 2, 45, 1, 1, 0),
 [AC6] 411: (0, 0, 2, 12, 65, 1, 32, 34, 13, 1, 1, 0)
 [F26] 412: (0, 0, 0, 1, 4, 10, 34, 3, 6, 42, 2, 0),
 [FV6] 413: (0, 14, 16, 33, 65, 1, 0, 2, 68, 56, 0, 0),
 [6B6] 414: (0, 6, 24, 97, 1, 1, 0, 2, 30, 97, 0, 0),
 [8T5] 415: (0, 0, 2, 4, 9, 3, 5, 25, 8, 0, 0, 0),
 [966] 416: (0, 0, 0, 1, 14, 48, 64, 0, 0, 64, 0, 0),
 [N67] 417: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 418: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 419: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 420: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 421: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 422: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 423: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 424: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 425: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 426: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 427: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 428: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [MY5] 429: (0, 15, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0),
 [AY5] 430: (0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 15, 0),
 [7T6] 431: (0, 64, 116, 0, 8, 0, 16, 0, 42, 6, 74, 0),
 [307] 432: (0, 64, 116, 0, 8, 0, 20, 0, 40, 14, 64, 0),
 [CS5] 433: (0, 0, 0, 0, 0, 95, 0, 0, 0, 0, 0, 0),
 [ZH6] 434: (0, 16, 0, 40, 0, 84, 0, 40, 0, 68, 0, 0),
 [AF6] 435: (0, 0, 68, 0, 40, 0, 84, 0, 40, 0, 16, 0),
 [M47] 436: (0, 146, 0, 0, 146, 0, 0, 146, 0, 0, 146, 0),
 [YM5] 437: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
 [AM7] 438: (34, 0, 136, 34, 0, 136, 34, 0, 136, 34, 0, 136),
 [7T5] 439: (0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0),
 [LV5] 440: (0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0),
 [7T5] 441: (0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0),
 [036] 442: (0, 0, 0, 0, 0, 255, 0, 0, 0, 255, 0, 0),
 [6N5] 443: (0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0),
 [N67] 444: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [D36] 445: (0, 0, 0, 0, 0, 249, 0, 0, 0, 255, 0, 0),
 [N67] 446: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 447: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 448: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 449: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 450: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 451: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 452: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 453: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 454: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),

[N67] 455: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 456: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 457: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 458: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 459: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 460: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 461: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 462: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 463: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 464: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 465: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 466: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 467: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 468: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 469: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 470: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 471: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 472: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 473: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 474: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 475: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 476: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 477: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 478: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [K1A] 479: (255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255),
 [N67] 480: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 481: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 482: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 483: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 484: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [CH6] 485: (0, 3, 12, 48, 65, 9, 8, 70, 48, 0, 0, 0),
 [N67] 486: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 487: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 488: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 489: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 490: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 491: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 492: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 493: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 494: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 495: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [N67] 496: (34, 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4),
 [DB7] 497: (24, 36, 66, 0, 32, 64, 0, 66, 36, 26, 64, 0),
 [SC6] 498: (34, 67, 1, 0, 30, 2, 64, 0, 48, 2, 0, 0),
 [UP6] 499: (24, 36, 66, 0, 8, 0, 4, 66, 36, 26, 0, 0),
 [AF6] 500: (80, 1, 1, 30, 0, 64, 88, 39, 1, 2, 0, 0)


```
[E87] 501: ( 34, 48, 17, 17, 62, 81, 16, 16, 2, 98, 0, 0 ),
[437] 502: ( 66, 67, 1, 64, 94, 2, 0, 32, 32, 2, 64, 0 ),
[B66] 503: ( 0, 126, 0, 1, 1, 1, 0, 126, 2, 2, 0, 0 ),
[X66] 504: ( 0, 0, 0, 3, 12, 48, 72, 0, 2, 13, 0, 0 ),
[YM6] 505: ( 126, 0, 1, 1, 14, 1, 1, 0, 126, 66, 0, 0 ),
[377] 506: ( 85, 1, 72, 62, 0, 60, 66, 16, 0, 98, 4, 0 ),
[YM5] 507: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[457] 508: ( 68, 2, 6, 16, 20, 26, 18, 50, 80, 4, 0, 0 ),
[366] 509: ( 1, 1, 1, 1, 1, 1, 255, 0, 0, 255, 0, 0 ),
[LB6] 510: ( 128, 64, 32, 16, 8, 4, 2, 1, 0, 0, 0, 0 ),
[LV5] 511: ( 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0 ),
[R87] 512: ( 0, 64, 0, 128, 0, 128, 0, 64, 0, 32, 0, 0 ),
[YM5] 513: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8 ),
[YM5] 514: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[8W5] 515: ( 0, 14, 8, 1, 1, 1, 8, 14, 2, 2, 0, 0 );
[000] 516:
[000] 517:
[000] 518:
[000] 519:
[XR9] 520: UNTENPUNKTE : ARRAY [ 1..255, 1..12 ]
OF BYTE =
[7M5] 521: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 522: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 523: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 524: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 525: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[8T5] 526: ( 0, 0, 0, 0, 0, 0, 128, 0, 0, 0, 0, 0 ),
[YM5] 527: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 528: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 529: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 530: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 531: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 532: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 533: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 534: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 535: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 536: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 537: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 538: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 539: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 540: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[5A6] 541: ( 0, 0, 128, 8, 8, 128, 0, 0, 0, 0, 0, 0 ),
[YM5] 542: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 543: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 544: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 545: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 546: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 547: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 548: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 549: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 550: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 551: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 552: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 553: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 554: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 555: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[5R5] 556: ( 0, 0, 0, 0, 0, 136, 0, 0, 0, 0, 0, 0 ),
[YM5] 557: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 558: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 559: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 560: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 561: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 562: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 563: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[7S5] 564: ( 0, 0, 0, 8, 128, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 565: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 566: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 567: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 568: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 569: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 570: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 571: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 572: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 573: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 574: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 575: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 576: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 577: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 578: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[7S5] 579: ( 0, 0, 0, 8, 128, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 580: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 )
```

[YM5]	581:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	)
[YM5]	582:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	)
[YM5]	583:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	)
[YM5]	584:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	)
[YM5]	585:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	)
[YM5]	586:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	)
[YM5]	587:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	)
[YM5]	588:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	)
[YM5]	589:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	)
[YM5]	590:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	)
[YM5]	591:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	)
[YM5]	592:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	)
[YM5]	593:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	)
[R36]	594:	(	0.	0.	0.	0.	128.	8.	128.	0.	0.	0.	0.	0.)	
[YM5]	595:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	596:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	597:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	598:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	599:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	600:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	601:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	602:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	603:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	604:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	605:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	606:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	607:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	608:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[Y66]	609:	(	0.	0.	128.	8.	8.	8.	128.	0.	0.	0.	0.	0.)	
[YM5]	610:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	611:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	612:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	613:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	614:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[E25]	615:	(	8.	8.	8.	8.	8.	8.	8.	8.	8.	8.	8.	8.)	
[YM5]	616:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	617:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	618:	(	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.)	
[YM5]	619:	(													


```

[YM5] 666: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 667: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 668: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 669: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 670: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 671: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[R76] 672: ( 0, 0, 0, 128, 8, 8, 128, 0, 0, 0, 0, 0, 0 ),
[YM5] 673: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 674: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[7R5] 675: ( 0, 0, 0, 0, 128, 0, 0, 0, 0, 0, 0, 0, 0 ),
[BX6] 676: ( 0, 8, 128, 128, 8, 8, 8, 128, 0, 0, 0, 0, 0 ),
[YM5] 677: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 678: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 679: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 680: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 681: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 682: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 683: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 684: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 685: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 686: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 687: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 688: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 689: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 690: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 691: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 692: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 693: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 694: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 695: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[857] 696: ( 0, 148, 0, 0, 148, 0, 0, 148, 0, 0, 148, 0, 148,
0 ),
[KK8] 697: ( 64, 144, 32, 64, 144, 32, 64, 144, 32, 64,
144, 32 ),
[P77] 698: ( 2, 240, 8, 2, 240, 8, 2, 240, 8, 2, 240,
8 ),
[7T5] 699: ( 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0 ),
[LV5] 700: ( 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0 ),
[7T5] 701: ( 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0 ),
[036] 702: ( 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 255, 0, 0 ),
[036] 703: ( 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 255, 0, 0 ),
[YM5] 704: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[036] 705: ( 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 255, 0, 0 ),
[YM5] 706: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 707: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 708: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 709: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 710: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 711: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 712: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 713: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 714: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 715: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 716: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 717: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 718: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 719: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 720: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 721: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 722: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 723: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 724: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 725: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 726: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 727: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 728: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 729: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 730: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 731: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 732: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 733: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 734: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 735: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 736: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 737: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 738: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[KIA] 739: ( 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255 ),
[YM5] 740: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 741: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 742: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 743: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 744: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[VS5] 745: ( 0, 136, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 746: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 747: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 748: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 749: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),

```

```

[YM5] 750: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 751: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 752: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 753: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 754: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 755: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 756: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[FV5] 757: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255 ),
[AR5] 758: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 17 ),
[YM5] 759: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 760: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[6N5] 761: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 762: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2 ),
[YM5] 763: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 764: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 765: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 766: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[YM5] 767: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[WP5] 768: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 51 ),
[KB6] 769: ( 0, 0, 0, 0, 0, 0, 255, 0, 0, 255, 0, 51 ),
[5S5] 770: ( 0, 0, 0, 0, 0, 0, 0, 0, 136, 0, 0, 0 ),
[376] 771: ( 0, 0, 8, 8, 8, 8, 136, 0, 0, 0, 51 ),
[YM5] 772: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[EY5] 773: ( 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0 ),
[YM5] 774: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ),
[9M5] 775: ( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 );

```

Listing 2. In dieser Datei stehen die Daten für den neuen Zeichensatz. Eine Änderung ist schon vorgegeben.

```

[MVB] 0: PROGRAM DRUCKE: ( *** 19.08.87 ** Günter Fri
tz *** )
[FD5] 1: TYPE NAME = STRING [ 66 ];
[HO6] 2: ( $I CONST.DTA ) ( DATEN )
[AK5] 3: VAR DATEI : FILE OF BYTE;
[UX7] 4: DRUCKZEILE : ARRAY [ 1..96 ] OF BYTE;
[X9A] 5: MS ( MASTER SELECT ) : ARRAY [ 0..96 ]
OF BYTE;
[WX5] 6: URZEILE, ZEILE, LINE : BYTE;
[D36] 7: ALAUF, BLAUF, CLAUF : INTEGER;
[Y47] 8: ZAEHL : ARRAY [ 0..4 ] OF INTEGER;
[U24] 9: AUSDRUCK : TEXT;
[4T3] 10: ZEICHEN : CHAR;
[YP3] 11: DRUCKDATEI : NAME;
[G4E] 12: GEDRUCKT, UNTERSTREICHE, FETT, HOCH, TI
EF, HT, BREIT : BOOLEAN;
[000] 13:
[E6A] 14: FUNCTION EXIST ( DATEINAME : NAME ) : BOOLE
AN;
[EE0] 15: VAR
[IR3] 16: PRUEFDATEI : FILE;
[B21] 17: BEGIN
[537] 18: ASSIGN ( PRUEFDATEI, DATEINAME );
[361] 19: ( $I- )
[B94] 20: RESET ( PRUEFDATEI );
[V51] 21: ( $I+ )
[DS5] 22: EXIST := ( IOresult = 0 );
[B84] 23: CLOSE ( PRUEFDATEI );
[WD3] 24: END; ( EXIST )
[000] 25:
[E9C] 26: PROCEDURE VORSCHUB ( ZAHL : BYTE ); ( IN 21
6tel '' )
[B21] 27: BEGIN
[N89] 28: WRITE ( LST, CHR ( 27 ), CHR ( 74 ), CHR
( ZAHL ));
[MR4] 29: ZEILE := ZEILE + ZAHL;
[LL7] 30: IF ZEILE = URZEILE THEN ZEILE := 0
[TM0] 31: END;

```



```

[000] 32:
[TY2] 33: PROCEDURE LF;
[C59] 34: BEGIN VORSCHUB ( URZEILE - ZEILE ) END;
[000] 35:
[D24] 36: PROCEDURE DRUCKE;
[2A2] 37: VAR PUFFER1
[3W1] 38: : BYTE;
[B21] 39: BEGIN
[FB4] 40: IF NOT GEDRUCKT THEN
[B21] 41: BEGIN
[FU6] 42: IF HT THEN ( INDEX/POTENZ )
[B21] 43: BEGIN
[KE6] 44: FOR ALAUF := 1 TO ZAEHL [ 0 ] DO
[1H6] 45: IF MS [ ALAUF ] AND 1 = 1 THEN
[B21] 46: BEGIN
[ER5] 47: WRITE ( LST, CHR ( 27 ), '
S', '0',
[356] 48: CHR ( DRUCKZEILE [
ALAUF ] ));
[SH5] 49: DRUCKZEILE [ ALAUF ] := 32
[DD0] 50: END
[OV6] 51: ELSE IF MS [ ALAUF ] AND 2 = 2
THEN
[B21] 52: BEGIN
[NR5] 53: WRITE ( LST, CHR ( 27 ), '
S', '1',
[356] 54: CHR ( DRUCKZEILE [
ALAUF ] ));
[SH5] 55: DRUCKZEILE [ ALAUF ] := 32
[DD0] 56: END
[EZ5] 57: ELSE WRITE ( LST, CHR ( 32 ));
[R93] 58: END; ( IF HT )
[CE4] 59: WRITE ( LST, CHR ( 13 ));
[N64] 60: FOR ALAUF := 1 TO 4 DO
[B21] 61: BEGIN
[VL2] 62: PUFFER1 := 0;
[WT6] 63: IF ZAEHL [ ALAUF ] > 0 THEN BEGI
N
[4F4] 64: WRITE ( LST, CHR ( 27 ));
[PD3] 65: WRITE ( LST, 'L' );
[R6A] 66: WRITE ( LST, CHR ( ( 12 * ZAEHL [
ALAUF ] ) MOD 256 ));
[FFB] 67: WRITE ( LST, CHR ( TRUNC ( 12 *
ZAEHL [ ALAUF ] / 256 ));
[7R2] 68: CASE ALAUF OF
[LS6] 69: 1: FOR BLAUF := 1 TO ZAEHL
[ 1 ] DO
[LP6] 70: IF MS [ BLAUF - 1 ] A
ND 32 = 0 THEN
[HK4] 71: FOR CLAUF := 1 TO 12
DO
[B21] 72: BEGIN
[6R6] 73: IF MS [ BLAUF ] A
ND 32 = 32 THEN
[B21] 74: BEGIN
[PE6] 75: WRITE ( LST,
CHR ( OBENPUNKTE1
[PG7] 76: [ DRUCKZEI
LE [ BLAUF ], CLAUF ] ));
[V35] 77: PUFFER1 := 0
BENPUNKTE1
[A77] 78: [ DRUCKZEILE [ BLAUF ], CLAUF ];
[2A3] 79: END; ( IF MS )
[PE6] 80: WRITE ( LST, CHR
( OBENPUNKTE1
[EX9] 81: [ DRUCKZEILE [
BLAUF ], CLAUF ] OR PUFFER1));
[CK6] 82: IF MS [ BLAUF ]
AND 8 = 8 THEN
[B21] 83: BEGIN
[V35] 84: PUFFER1 := 0
BENPUNKTE1
[A77] 85: [ DRUCKZEILE [ BLAUF ], CLAUF ];
[F66] 86: END ( SCHATTEN
SCHRIFT )
[MM3] 87: ELSE PUFFER1 :=
0;
[HY3] 88: END; ( CLAUF )
[US6] 89: 2: FOR BLAUF := 1 TO ZAEHL
[ 2 ] DO
[LP6] 90: IF MS [ BLAUF - 1 ] A
ND 32 = 0 THEN
[HK4] 91: FOR CLAUF := 1 TO 12
DO
[B21] 92: BEGIN
[6R6] 93: IF MS [ BLAUF ]
AND 32 = 32 THEN
[B21] 94: BEGIN
[WE6] 95: WRITE ( LST,
CHR ( OBENPUNKTE2
[PG7] 96: [ DRUCKZEI
LE [ BLAUF ], CLAUF ] ));

```

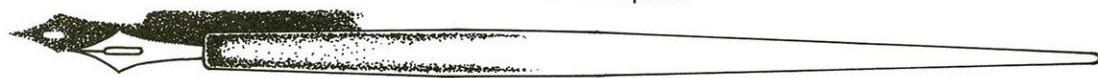
```

[145] 97: PUFFER1 := 0
BENPUNKTE2
[A77] 98: [ DRUCKZEILE [ BLAUF ], CLAUF ];
[2A3] 99: END; ( IF MS )
[WE6] 100: WRITE ( LST, CHR
( OBENPUNKTE2
[V4A] 101: [ DRUCKZEILE
[ BLAUF ], CLAUF ] OR PUFFER1 ));
[CK6] 102: IF MS [ BLAUF ]
AND 8 = 8 THEN
[B21] 103: BEGIN
[145] 104: PUFFER1 := 0
BENPUNKTE2
[A77] 105: [ DRUCKZEILE [ BLAUF ], CLAUF ];
[F66] 106: END ( SCHATTEN
SCHRIFT )
[YY1] 107: ELSE BEGIN
[VL2] 108: PUFFER1 :
= 0;
[TM0] 109: END;
[EB3] 110: END; ( CLAUF )
[1T6] 111: 3: FOR BLAUF := 1 TO ZAEHL
[ 3 ] DO
[LP6] 112: IF MS [ BLAUF - 1 ] A
ND 32 = 0 THEN
[HK4] 113: FOR CLAUF := 1 TO 12
DO
[B21] 114: BEGIN
[P19] 115: IF MS [ BLAUF ]
AND 128 = 128 THEN LINE := 8
[N63] 116: ELSE LINE := 0;
[6R6] 117: IF MS [ BLAUF ]
AND 32 = 32 THEN
[B21] 118: BEGIN
[VV8] 119: WRITE ( LST,
CHR ( ( TRUNC ( ( UNTENPUNKTE
[YN8] 120: [ DRUCKZEIL
E [ BLAUF ], CLAUF ] AND 240 )
[2B3] 121: / 16 ) ) OR LINE ));
[6U7] 122: PUFFER1 := T
RUNC ( ( UNTENPUNKTE
[YN8] 123: [ DRUCKZEIL
E [ BLAUF ], CLAUF ] AND 240 )
[GV0] 124: / 16 )
[2A3] 125: END; ( IF MS
)
[159] 126: WRITE ( LST, CHR
( TRUNC ( ( UNTENPUNKTE
[4T9] 127: [ DRUCKZEILE [ B
LAUF ], CLAUF ] AND 240 ) / 16 )
[LM4] 128: OR PUFFER1 OR LINE ));
[365] 129: IF MS [ BLAUF ]
AND 8 = 8
[UW7] 130: THEN PUFFER1 :
= TRUNC ( ( UNTENPUNKTE
[4T9] 131: [ DRUCKZEILE [
BLAUF ], CLAUF ] AND 240 ) / 16 )
[MM3] 132: ELSE PUFFER1 :=
0;
[EB3] 133: END; ( CLAUF )
[MC7] 134: 4: FOR BLAUF := 1 TO ( ZAEH
L [ 4 ] ) DO
[LP6] 135: IF MS [ BLAUF - 1 ] A
ND 32 = 0 THEN
[HK4] 136: FOR CLAUF := 1 TO 12
DO
[B21] 137: BEGIN
[6R6] 138: IF MS [ BLAUF ]
AND 32 = 32 THEN
[B21] 139: BEGIN
[PR6] 140: WRITE ( LST,
CHR ( UNTENPUNKTE
[5P8] 141: [ DRUCKZEI
LE [ BLAUF ], CLAUF ] AND 15 ));
[DN5] 142: PUFFER1 := (
UNTENPUNKTE
[6H8] 143: [ DRUCKZE
ILE [ BLAUF ], CLAUF ] AND 15 )
[2A3] 144: END; ( IF MS
)
[767] 145: WRITE ( LST, CHR
( ( UNTENPUNKTE
[6H8] 146: [ DRUCKZEILE
[ BLAUF ], CLAUF ] AND 15 )
[HX2] 147: OR PUFFER1 ));
[365] 148: IF MS [ BLAUF ]
AND 8 = 8
[DL6] 149: THEN PUFFER1 :

```


CONTEXT PC

Autor: Matthias Uphoff



Diese Textverarbeitung zeigt, was sie leistet!

Was Context PC nicht kann:

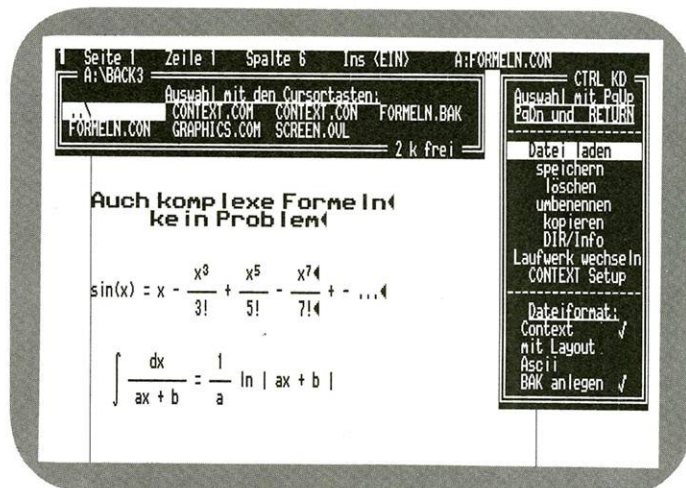
- Dokumente mit 4000 Seiten verwalten
- 3-fach geschachtelte Fußnoten
- Grafik aus Fremdprogrammen einbinden.

Wenn Sie diese Features unbedingt benötigen, empfehlen wir Ihnen die Textverarbeitung der Firma XXX!

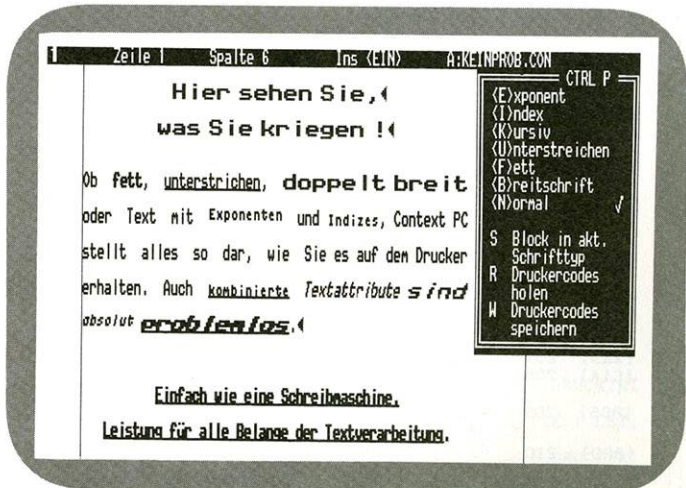
Was Context PC dafür jedoch

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



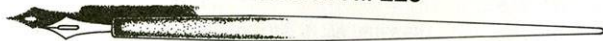
Der Dateimanager des Context PC macht DOS-Benutzeroberflächen überflüssig.



»WYSIWYG« in Reinform: Alle Schriftattribute werden am Bildschirm dargestellt.

CONTEXT PC

Bestell-Nr. 228



DM 99,-

unverbindliche Preisempfehlung

- Bitte Bestellkarte benutzen -

DMV GmbH · Abt. Software · Postfach 250 · 3440 Eschwege

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


```

= ( UNTENPUNKTE
[6H8] 150: [ DRUCKZE
ILE [ BLAUF ], CLAUF ] AND 15 )
[MM3] 151: ELSE PUFFER1 :=
0;
[EB3] 152: END; ( CLAUF )
[2X2] 153: END; ( CASE )
[CE4] 154: WRITE ( LST, CHR ( 13 ) );
[DT3] 155: END; ( IF ZAEHL )
[VC5] 156: IF ZAEHL [ ALAUF ] > 1 THEN
[7R2] 157: CASE ALAUF OF
[734] 158: 1, 3 : VORSCHUB ( 1 );
[CA4] 159: 2 : VORSCHUB ( 11 );
[2X2] 160: END; ( CASE )
[EB3] 161: END; ( ALAUF )
[4K4] 162: END; ( IF GEDRUCKT )
[062] 163: HT := FALSE;
[PO4] 164: GEDRUCKT := TRUE;
[GV3] 165: END; ( DRUCKE )
[000] 166:
[B21] 167: BEGIN
[FY1] 168: ZEILE := 0;
[343] 169: URZEILE := 36;
[1P5] 170: UNTERSTREICHE := FALSE;
[CK3] 171: BREIT := FALSE;
[P33] 172: FETT := FALSE;
[913] 173: HOCH := FALSE;
[713] 174: TIEF := FALSE;
[062] 175: HT := FALSE;
[G58] 176: FOR ALAUF := 0 TO 4 DO ZAEHL [ ALAUF ] := 0;

[WB2] 177: MS [ 0 ] := 0;
[NZ1] 178: CLRSCR;
[S6C] 179: WRITELN ( 'BITTE DIE ZU DRUCKENDE DATEI EIN
GEBEN' );
[6T3] 180: READ ( DRUCKDATEI );
[ZOC] 181: WHILE NOT EXIST ( DRUCKDATEI ) AND ( DRUCKD
ATEI <> '' ) DO
[B21] 182: BEGIN
[N63] 183: WRITE ( CHR ( 7 ) );
[TD4] 184: GOTOXY ( 1, WHEREY );
[602] 185: DELLINE;
[6T3] 186: READ ( DRUCKDATEI );
[VA3] 187: END; ( WHILE )
[BG6] 188: IF DRUCKDATEI = '' THEN EXIT;
[X97] 189: ASSIGN ( AUSDRUCK, DRUCKDATEI );
[9Z3] 190: RESET ( AUSDRUCK );
[BX3] 191: GEDRUCKT := FALSE;
[NZ1] 192: CLRSCR;
[XG1] 193: REPEAT
[2K5] 194: READ ( AUSDRUCK, ZEICHEN );
[9R3] 195: WRITE ( ZEICHEN );
[F75] 196: CASE ORD ( ZEICHEN ) OF
[DP1] 197: 10 : BEGIN
[5X1] 198: DRUCKE;
[9C0] 199: LF;
[272] 200: END; ( 10 )
[LP1] 201: 13 : BEGIN
[5X1] 202: DRUCKE;
[G58] 203: FOR ALAUF := 0 TO 4 DO ZAEHL [ ALA
UF ] := 0;
[572] 204: END; ( 13 )
[L92] 205: 26 : DRUCKE;
[WP1] 206: 27 : BEGIN
[2K5] 207: READ ( AUSDRUCK, ZEICHEN );
[C1A] 208: IF ZEICHEN IN [ '-', 'E', 'H', 'O'
, '3', 'A', 'S', 'T', 'W', 's', 'U', 'J', '!'
, 'x', 'R' ]
[8D5] 209: THEN
[BR0] 210: CASE ZEICHEN OF
[FN3] 211: 's', 'U', 'J', '!', 'x', 'R' : (
Verden ignoriert )
[ANB] 212:
[2K5] 213: RE
AD ( AUSDRUCK, ZEICHEN );
[MN2] 214: : BEGIN
[2K5] 215: READ ( AUSDRUCK, ZE
ICHEN );
[F75] 216: CASE ORD ( ZEICHEN
) OF
[1K6] 217: 1, 49 : UNTERSTRE
ICHE := TRUE;
[7Z6] 218: 0, 48 : UNTERSTRE
ICHE := FALSE;
[2X2] 219: END; ( CASE )
[NO2] 220: END; ( - )
[T65] 221: 'E', 'G' : FETT := TRUE;
[W55] 222: 'F', 'H' : FETT := FALSE;
[5A4] 223: 'O' : URZEILE := 27;
[7A4] 224: '1' : URZEILE := 21;
[GA4] 225: '2' : URZEILE := 36;
[OP2] 226: '3' : BEGIN
[2K5] 227: READ ( AUSDRUCK, ZE
ICHEN );

```

```

[1V5] 228: URZEILE := ORD ( ZE
ICHEN );
[NO2] 229: END; ( 3 )
[WP2] 230: 'A' : BEGIN
[2K5] 231: READ ( AUSDRUCK, ZE
ICHEN );
[7T6] 232: URZEILE := ORD ( ZE
ICHEN ) * 3;
[NO2] 233: END; ( A )
[OS2] 234: 'S' : BEGIN
[2K5] 235: READ ( AUSDRUCK, ZE
ICHEN );
[F75] 236: CASE ORD ( ZEICHEN
) OF
[9D9] 237: 1, 49 : BEGIN HOC
H := FALSE; TIEF := TRUE; END;
[TE9] 238: 0, 48 : BEGIN TIE
F := FALSE; HOCH := TRUE; END;
[2X2] 239: END; ( CASE )
[NO2] 240: END; ( S )
[2S2] 241: 'T' : BEGIN
[913] 242: HOCH := FALSE;
[713] 243: TIEF := FALSE;
[NO2] 244: END; ( T )
[8S2] 245: 'W' : BEGIN
[2K5] 246: READ ( AUSDRUCK, ZE
ICHEN );
[F75] 247: CASE ORD ( ZEICHEN
) OF
[CE4] 248: 1, 49 : BREIT :=
TRUE;
[KV4] 249: 0, 48 : BREIT :=
FALSE;
[2X2] 250: END; ( CASE )
[NO2] 251: END; ( W )
[2X2] 252: END; ( CASE )
[972] 253: END; ( 27 )
[51A] 254: 1..6, ( ALLE DRUCKBA
REN ZEICHEN )
[190] 255: 16,
[P91] 256: 21..23,
[X91] 257: 28..31,
[MB1] 258: 32..126,
[S33] 259: 128..254 : BEGIN
[260] 260: GEDRUCKT := FALSE;
[756] 261: ZAEHL [ 0 ] := ZAEHL [ 0 ] +
1;
[6Z3] 262: IF UNTERSTREICHE
[3HA] 263: THEN MS [ ZAEHL [ 0 ] ] :=
MS [ ZAEHL [ 0 ] ] OR 128
[42B] 264: ELSE MS [ ZAEHL [ 0 ] ] := MS
[ ZAEHL [ 0 ] ] AND 127;
[MV1] 265: IF BREIT
[HDA] 266: THEN MS [ ZAEHL [ 0 ] ] :=
MS [ ZAEHL [ 0 ] ] OR 40
[X1B] 267: ELSE MS [ ZAEHL [ 0 ] ] :=
MS [ ZAEHL [ 0 ] ] AND 215;
[AZ1] 268: IF FETT
[1CA] 269: THEN MS [ ZAEHL [ 0 ] ] :=
MS [ ZAEHL [ 0 ] ] OR 8
[92B] 270: ELSE MS [ ZAEHL [ 0 ] ] := MS
[ ZAEHL [ 0 ] ] AND 247;
[NK2] 271: IF HOCH THEN
[B21] 272: BEGIN
[5Z8] 273: MS [ ZAEHL [ 0 ] ] := MS
[ ZAEHL [ 0 ] ] OR 1;
[502] 274: HT := TRUE;
[DD0] 275: END
[22B] 276: ELSE MS [ ZAEHL [ 0 ] ] := MS
[ ZAEHL [ 0 ] ] AND 254;
[5L2] 277: IF TIEF THEN
[B21] 278: BEGIN
[8Z8] 279: MS [ ZAEHL [ 0 ] ] := MS
[ ZAEHL [ 0 ] ] OR 2;
[MU1] 280: HT := TRUE
[DD0] 281: END
[Z1B] 282: ELSE MS [ ZAEHL [ 0 ] ] := MS
[ ZAEHL [ 0 ] ] AND 253;
[WK9] 283: DRUCKZEILE [ ZAEHL [ 0 ] ] :
= ORD ( ZEICHEN );
[7K4] 284: FOR ALAUF := 1 TO 12 DO
[B21] 285: BEGIN
[6F8] 286: IF OBENPUNKTE1 [ ORD (
ZEICHEN ), ALAUF ]
[SE0] 287: <> 0
[EG6] 288: THEN ZAEHL [ 1 ] :=
ZAEHL [ 0 ];
[CF8] 289: IF OBENPUNKTE2 [ ORD (
ZEICHEN ), ALAUF ]
[SE0] 290: <> 0
[MG6] 291: THEN ZAEHL [ 2 ] :=

```



```

[COB] 292: ZAEHL [ 0 ];
[NE7] 293: IF ( TRUNC ( UNTENPUNKT
          E [ ORD ( ZEICHEN ), ALAUF ]
          ) / 16 <> 0 ) OR UNTERSTREICHE
          AND 240
[UG6] 294: THEN ZAEHL [ 3 ] :=
          ZAEHL [ 0 ];
[EVA] 295: IF UNTENPUNKTE [ ORD (
          ZEICHEN ), ALAUF ] AND 15 <> 0
[VK6] 296: THEN ZAEHL [ 4 ] :=
          ZAEHL [ 0 ];
[EB3] 297: END; ( ALAUF )
[EX2] 298: IF BREIT THEN
[B21] 299: BEGIN
[N64] 300: FOR ALAUF := 1 TO 4 DO
[4S6] 301: IF ZAEHL [ ALAUF ] = ZA
          EHL [ 0 ]
[RD9] 302: THEN ZAEHL [ ALAUF ]
          := ZAEHL [ ALAUF ] + 1;
[T56] 303: ZAEHL [ 0 ] := ZAEHL [ 0 ]
          + 1;
[AH4] 304: MS [ ZAEHL [ 0 ] ] := 0;
[TMO] 305: END;
[BN3] 306: END; ( 1..6, ..... )
[2X2] 307: END; ( CASE )
[M85] 308: UNTIL ORD ( ZEICHEN ) = 26;
[5LO] 309: END.

```

Listing 3. Quellcode des Druckprogramms für alle Epson-kompatiblen Matrixdrucker

```

[MVB] 0: PROGRAM DRUCKE; (** 19.08.87 ** Günter Fri
      tz **)
[FD5] 1: TYPE NAME = STRING [ 66 ];
[H06] 2: ($I CONST.DTA ) ( DATEN )
[AK5] 3: VAR DATEI : FILE OF BYTE;
[UX7] 4: DRUCKZEILE : ARRAY [ 1..96 ] OF BYTE;
[X9A] 5: MS ( MASTER SELECT ) : ARRAY [ 0..96 ]
      OF BYTE;
[WX5] 6: URZEILE, ZEILE, LINE : BYTE;
[D36] 7: ALAUF, BLAUF, CLAUF : INTEGER;
[Y47] 8: ZAEHL : ARRAY [ 0..4 ] OF INTEGER;
[U24] 9: AUSDRUCK : TEXT;
[4T3] 10: ZEICHEN : CHAR;
[YP3] 11: DRUCKDATEI : NAME;
[G4E] 12: GEDRUCKT, UNTERSTREICHE, FETT, HOCH, TI
      EF, HT, BREIT : BOOLEAN;
[000] 13:
[E6A] 14: FUNCTION EXIST ( DATEINAME : NAME ) : BOOLE
      AN;
[EE0] 15: VAR
[1R3] 16: PRUEFDATEI : FILE;
[B21] 17: BEGIN
[537] 18: ASSIGN ( PRUEFDATEI, DATEINAME );
[361] 19: ($I-)
[B94] 20: RESET ( PRUEFDATEI );
[V51] 21: ($I+)
[DS5] 22: EXIST := ( IOresult = 0 );
[B84] 23: CLOSE ( PRUEFDATEI );
[WD3] 24: END; ( EXIST )
[000] 25:
[E9C] 26: PROCEDURE VORSCHUB ( ZAHL : BYTE ); ( IN 21
      6tel '' )
[B21] 27: BEGIN
[N89] 28: WRITE ( LST, CHR ( 27 ), CHR ( 74 ), CHR
      ( ZAHL ) );
[MR4] 29: ZEILE := ZEILE + ZAHL;
[LL7] 30: IF ZEILE = URZEILE THEN ZEILE := 0
[TM0] 31: END;
[000] 32:
[TY2] 33: PROCEDURE LF;
[C59] 34: BEGIN VORSCHUB ( URZEILE - ZEILE ) END;
[000] 35:
[D24] 36: PROCEDURE DRUCKE;
[2A2] 37: VAR PUFFER1
[7Y1] 38: , PUFFER2

```

```

[3W1] 39: : BYTE;
[B21] 40: BEGIN
[FH4] 41: IF NOT GEDRUCKT THEN
[B21] 42: BEGIN
[FU6] 43: IF HT THEN ( INDEX/POTENZ )
[B21] 44: BEGIN
[KE6] 45: FOR ALAUF := 1 TO ZAEHL [ 0 ] DO
[1H6] 46: IF MS [ ALAUF ] AND 1 = 1 THEN
[B21] 47: BEGIN
[ER5] 48: WRITE ( LST, CHR ( 27 ), '
      S', 'O',
[356] 49: CHR ( DRUCKZEILE [
      ALAUF ] ) );
[SH5] 50: DRUCKZEILE [ ALAUF ] := 32
[DD0] 51: END
[OV6] 52: ELSE IF MS [ ALAUF ] AND 2 = 2
      THEN
[B21] 53: BEGIN
[NR5] 54: WRITE ( LST, CHR ( 27 ), '
      S', '1',
[356] 55: CHR ( DRUCKZEILE [
      ALAUF ] ) );
[SH5] 56: DRUCKZEILE [ ALAUF ] := 32
[DD0] 57: END
[EZ5] 58: ELSE WRITE ( LST, CHR ( 32 ) );
[R93] 59: END; ( IF HT )
[CE4] 60: WRITE ( LST, CHR ( 13 ) );
[N64] 61: FOR ALAUF := 1 TO 4 DO
[B21] 62: BEGIN
[VL2] 63: PUFFER1 := 0;
[2M2] 64: PUFFER2 := 0;
[2U6] 65: IF ZAEHL [ ALAUF ] > 1 THEN BEGI
      N
[7R2] 66: CASE ALAUF OF
[1X1] 67: 1..2 : BEGIN
[4F4] 68: WRITE ( LST, CHR ( 27
      ) );
[MH3] 69: WRITE ( LST, ' ' );
[8B4] 70: WRITE ( LST, CHR ( 1
      ) );
[TM0] 71: END;
[BX1] 72: 3..4 : BEGIN
[4F4] 73: WRITE ( LST, CHR ( 27
      ) );
[PD3] 74: WRITE ( LST, 'L' );
[TM0] 75: END;
[2X2] 76: END; ( CASE )
[R6A] 77: WRITE ( LST, CHR ( ( 12 * ZAEHL [
      ALAUF ] ) MOD 256 ) );
[FFB] 78: WRITE ( LST, CHR ( TRUNC ( 12 *
      ZAEHL [ ALAUF ] / 256 ) ) );
[7R2] 79: CASE ALAUF OF
[LS6] 80: 1: FOR BLAUF := 1 TO ZAEHL
      [ 1 ] DO
[LP6] 81: IF MS [ BLAUF - 1 ] A
      ND 32 = 0 THEN
[HK4] 82: FOR CLAUF := 1 TO 12
      DO
[B21] 83: BEGIN
[6R6] 84: IF MS [ BLAUF ] A
      ND 32 = 32 THEN
[B21] 85: BEGIN
[PE6] 86: WRITE ( LST,
      CHR ( OBENPUNKTE1
      [ DRUCKZEI
      LE [ BLAUF ], CLAUF ] ) );
[PG7] 87:
[PR6] 88: WRITE ( LST,
      CHR ( UNTENPUNKTE
      [ DRUCKZEI
      LE [ BLAUF ], CLAUF ]
      AND 128 ) )
[VR1] 90:
[V35] 91: PUFFER1 := 0
      BENPUNKTE1
[A77] 92: [ DRUCKZEILE [ BLAUF ], CLAUF ];
[2D5] 93: PUFFER2 := U
      NTENPUNKTE
[WN8] 94: [ DRUCKZEILE
      [ BLAUF ], CLAUF ] AND 128
[2A3] 95: END; ( IF MS )
[PE6] 96: WRITE ( LST, CHR
      ( OBENPUNKTE1
      [ DRUCKZEILE [
      BLAUF ], CLAUF ] OR PUFFER1 ) );
[EX9] 97:
[767] 98: WRITE ( LST, CHR
      ( ( UNTENPUNKTE
      [ DRUCKZEILE [
      BLAUF ], CLAUF ]
      AND 128 ) OR PU
      FFER2 ) );
[CK6] 101: IF MS [ BLAUF ]
      AND 8 = 8 THEN

```



```

[B21] 102: BEGIN
[V35] 103: PUFFER1 := 0
[BENPUNKTE1]
[A77] 104: [ DRUCKZEILE [ BLAUF ], CLAUF ];
[2D5] 105: PUFFER2 := U
[RR8] 106: [ DRUCKZEILE
[F66] 107: [ BLAUF ], CLAUF ] AND 128 ;
[YY1] 108: SCHRIFT )
[VL2] 109: ELSE BEGIN
= 0; PUFFER1 :
[VA2] 110: PUFFER2 :
= 0
[TM0] 111: END;
[HY3] 112: END; ( CLAUF )
[US6] 113: 2: FOR BLAUF := 1 TO ZAEHL
[ 2 ] DO
[LP6] 114: IF MS [ BLAUF - 1 ] A
[HK4] 115: ND 32 = 0 THEN
DO
FOR CLAUF := 1 TO 12
DO
[B21] 116: BEGIN
[5L9] 117: IF MS [ BLAUF ]
AND 128 = 128 THEN LINE := 255
[N63] 118: ELSE LINE := 0;
[6R6] 119: IF MS [ BLAUF ]
AND 32 = 32 THEN
[B21] 120: BEGIN
[WE6] 121: WRITE ( LST,
CHR ( OBENPUNKTE2
[PG7] 122: [ DRUCKZEI
LE [ BLAUF ], CLAUF ] );
[6M6] 123: WRITE ( LST,
CHR ( (( UNTEPUNKTE
[NB9] 124: [ DRUCKZEILE
[ BLAUF ], CLAUF ] AND 8 ) * 16 )
[62] 125: OR LINE );
PUFFER1 := 0
[145] 126: BENPUNKTE2
[A77] 127: [ DRUCKZEILE [ BLAUF ], CLAUF ];
[DF5] 128: PUFFER2 := (
[ UNTEPUNKTE
[A77] 129: [ DRUCKZEIL
E [ BLAUF ], CLAUF ]
[8H2] 130: AND 8 ) * 16 );
[2A3] 131: END; ( IF MS )
[WE6] 132: WRITE ( LST, CHR
( OBENPUNKTE2
[V4A] 133: [ DRUCKZEILE
[ BLAUF ], CLAUF ] OR PUFFER1 );
[6M6] 134: WRITE ( LST, CH
R ( (( UNTEPUNKTE
[NB9] 135: [ DRUCKZEILE [ B
LAUF ], CLAUF ] AND 8 ) * 16 )
[U4] 136: OR LINE OR PUFFER2 );
[CK6] 137: IF MS [ BLAUF ]
AND 8 = 8 THEN
[B21] 138: BEGIN
[145] 139: PUFFER1 := 0
BENPUNKTE2
[A77] 140: [ DRUCKZEILE [ BLAUF ], CLAUF ];
[DF5] 141: PUFFER2 := (
[ UNTEPUNKTE
[A77] 142: [ DRUCKZEIL
E [ BLAUF ], CLAUF ]
[8H2] 143: AND 8 ) * 16 );
[F66] 144: END ( SCHRIFT )
[YY1] 145: ELSE BEGIN
[VL2] 146: PUFFER1 :
= 0;
[VA2] 147: PUFFER2 :
= 0
[TM0] 148: END;
[EB3] 149: END; ( CLAUF )
[IT6] 150: 3: FOR BLAUF := 1 TO ZAEHL
[ 3 ] DO
[LP6] 151: IF MS [ BLAUF - 1 ] A
[HK4] 152: ND 32 = 0 THEN
DO
FOR CLAUF := 1 TO 12
DO
[B21] 153: BEGIN
[6R6] 154: IF MS [ BLAUF ]
AND 32 = 32 THEN
[B21] 155: BEGIN

```

```

[159] 156: WRITE ( LST,
[NN8] 157: CHR ( TRUNC ( ( UNTEPUNKTE
E [ BLAUF ], CLAUF ] AND 112 )
[A61] 158: / 16 );
[6U7] 159: PUFFER1 := T
[NN8] 160: [ DRUCKZEIL
E [ BLAUF ], CLAUF ] AND 112 )
[GVO] 161: / 16 )
[2A3] 162: END; ( IF MS
[159] 163: WRITE ( LST, CHR
( TRUNC ( ( UNTEPUNKTE
[VS9] 164: [ DRUCKZEILE [ B
LAUF ], CLAUF ] AND 112 ) / 16 )
[HX2] 165: OR PUFFER1 );
[365] 166: IF MS [ BLAUF ]
AND 8 = 8
[UW7] 167: THEN PUFFER1 :
= TRUNC ( ( UNTEPUNKTE
[VS9] 168: [ DRUCKZEILE [
BLAUF ], CLAUF ] AND 112 ) / 16 )
[MM3] 169: ELSE PUFFER1 :=
0;
[EB3] 170: END; ( CLAUF )
[MC7] 171: 4: FOR BLAUF := 1 TO ( ZAEH
L [ 4 ] ) DO
[LP6] 172: IF MS [ BLAUF - 1 ] A
[HK4] 173: ND 32 = 0 THEN
DO
FOR CLAUF := 1 TO 12
DO
[B21] 174: BEGIN
[6R6] 175: IF MS [ BLAUF ]
AND 32 = 32 THEN
[B21] 176: BEGIN
[PR6] 177: WRITE ( LST,
CHR ( UNTEPUNKTE
[OK8] 178: [ DRUCKZE
ILE [ BLAUF ], CLAUF ] AND 7 );
[DN5] 179: PUFFER1 := (
UNTEPUNKTE
[5G8] 180: [ DRUCKZE
ILE [ BLAUF ], CLAUF ] AND 7 )
[2A3] 181: END; ( IF MS
[767] 182: WRITE ( LST, CHR
[5G8] 183: [ DRUCKZEILE
[ BLAUF ], CLAUF ] AND 7 )
[HX2] 184: OR PUFFER1 );
[365] 185: IF MS [ BLAUF ]
AND 8 = 8
[DL6] 186: THEN PUFFER1 :
= ( UNTEPUNKTE
[5G8] 187: [ DRUCKZE
ILE [ BLAUF ], CLAUF ] AND 7 )
[MM3] 188: ELSE PUFFER1 :=
0;
[EB3] 189: END; ( CLAUF )
[2X2] 190: END; ( CASE )
[CE4] 191: WRITE ( LST, CHR ( 13 );
[DT3] 192: END; ( IF ZAEHL )
[VC5] 193: IF ZAEHL [ ALAUF ] > 1 THEN
[7R2] 194: CASE ALAUF OF
[734] 195: 1, 3 : VORSCHUB ( 1 );
[CA4] 196: 2 : VORSCHUB ( 11 );
[2X2] 197: END; ( CASE )
[EB3] 198: END; ( ALAUF )
[4K4] 199: END; ( IF GEDRUCKT )
[062] 200: HT := FALSE;
[PO4] 201: GEDRUCKT := TRUE;
[CV3] 202: END; ( DRUCKE )
[000] 203:
[B21] 204: BEGIN
[FY1] 205: ZEILE := 0;
[343] 206: URZEILE := 36;
[IP5] 207: UNTERSTREICHE := FALSE;
[CK3] 208: BREIT := FALSE;
[P33] 209: FETT := FALSE;
[913] 210: HOCH := FALSE;
[713] 211: TIEF := FALSE;
[062] 212: HT := FALSE;
[G58] 213: FOR ALAUF := 0 TO 4 DO ZAEHL [ ALAUF ] := 0
;
[WB2] 214: MS [ 0 ] := 0;
[NZ1] 215: CLRSCR;
[S6C] 216: WRITELN ( 'BITTE DIE ZU DRUCKENDE DATEI EIN
GEBEN' );

```



```

[6T3] 217: READ ( DRUCKDATEI );
[ZOC] 218: WHILE NOT EXIST ( DRUCKDATEI ) AND ( DRUCKD
ATEI <> '' ) DO
[B21] 219: BEGIN
[N63] 220: WRITE ( CHR ( 7 ) );
[TD4] 221: GOTOXY ( 1, WHEREY );
[602] 222: DELLINE;
[6T3] 223: READ ( DRUCKDATEI );
[VA3] 224: END; ( WHILE )
[BG6] 225: IF DRUCKDATEI = '' THEN EXIT;
[X97] 226: ASSIGN ( AUSDRUCK, DRUCKDATEI );
[9Z3] 227: RESET ( AUSDRUCK );
[BX3] 228: GEDRUCKT := FALSE;
[NZ1] 229: CLRSCR;
[XG1] 230: REPEAT
[2K5] 231: READ ( AUSDRUCK, ZEICHEN );
[9R3] 232: WRITE ( ZEICHEN );
[F75] 233: CASE ORD ( ZEICHEN ) OF
[DP1] 234: 10 : BEGIN
[5X1] 235: DRUCKE;
[9CO] 236: LF;
[272] 237: END; ( 10 )
[LP1] 238: 13 : BEGIN
[5X1] 239: DRUCKE;
[G58] 240: FOR ALAUF := 0 TO 4 DO ZAEHL [ ALA
UF ] := 0;
[572] 241: END; ( 13 )
[L92] 242: 26 : DRUCKE;
[WPI] 243: 27 : BEGIN
[2K5] 244: READ ( AUSDRUCK, ZEICHEN );
[C1A] 245: IF ZEICHEN IN [ '-', 'E', 'H', 'O',
'3', 'A', 'S', 'T', 'W',
's', 'U', 'J', 'I',
'x', 'R' ]
[BR0] 247: THEN
[FN3] 248: CASE ZEICHEN OF
[ANB] 249: 's', 'U', 'J', 'I', 'x', 'R' : (
Werden ignoriert )
[2K5] 250: RE
AD ( AUSDRUCK, ZEICHEN );
[MN2] 251: '-': BEGIN
[2K5] 252: READ ( AUSDRUCK, ZE
ICHEN );
[F75] 253: CASE ORD ( ZEICHEN
) OF
[1K6] 254: 1, 49 : UNTERSTRE
ICHE := TRUE;
[7Z6] 255: 0, 48 : UNTERSTRE
ICHE := FALSE;
[2X2] 256: END; ( CASE )
[N02] 257: END; ( - )
[T65] 258: 'E', 'G' : FETT := TRUE;
[W55] 259: 'F', 'H' : FETT := FALSE;
[5A4] 260: 'O' : URZEILE := 27;
[7A4] 261: '1' : URZEILE := 21;
[GA4] 262: '2' : URZEILE := 36;
[OP2] 263: '3' : BEGIN
[2K5] 264: READ ( AUSDRUCK, ZE
ICHEN );
[1V5] 265: URZEILE := ORD ( ZE
ICHEN );
[N02] 266: END; ( 3 )
[WP2] 267: 'A' : BEGIN
[2K5] 268: READ ( AUSDRUCK, ZE
ICHEN );
[7T6] 269: URZEILE := ORD ( ZE
ICHEN ) * 3;
[N02] 270: END; ( A )
[OS2] 271: 'S' : BEGIN
[2K5] 272: READ ( AUSDRUCK, ZE
ICHEN );
[F75] 273: CASE ORD ( ZEICHEN
) OF
[9D9] 274: 1, 49 : BEGIN HOC
H := FALSE; TIEF := TRUE END;
[TE9] 275: 0, 48 : BEGIN TIE
F := FALSE; HOCH := TRUE END;
[2X2] 276: END; ( CASE )
[N02] 277: END; ( S )
[2S2] 278: 'T' : BEGIN
[913] 279: HOCH := FALSE;
[713] 280: TIEF := FALSE;
[N02] 281: END; ( T )
[8S2] 282: 'W' : BEGIN
[2K5] 283: READ ( AUSDRUCK, ZE
ICHEN );
[F75] 284: CASE ORD ( ZEICHEN
) OF
[CE4] 285: 1, 49 : BREIT :=
TRUE;
[KV4] 286: 0, 48 : BREIT :=
FALSE;
[2X2] 287: END; ( CASE )

```

```

[N02] 288: END; ( W )
[2X2] 289: END; ( CASE )
[972] 290: END; ( 27 )
[51A] 291: 1..6, ( ALLE DRUCKBA
REN ZEICHEN )
[190] 292: 16,
[P91] 293: 21..23,
[X91] 294: 28..31,
[MB1] 295: 32..126,
[S33] 296: 128..254 : BEGIN
[T56] 297: ZAEHL [ 0 ] := ZAEHL [ 0 ] + 1;
[6Z3] 298: IF UNTERSTREICHE
[3HA] 299: THEN MS [ ZAEHL [ 0 ] ] := MS
[ ZAEHL [ 0 ] ] OR 128
[42B] 300: ELSE MS [ ZAEHL [ 0 ] ] := MS [
ZAEHL [ 0 ] ] AND 127;
[MV1] 301: IF BREIT
[HD4] 302: THEN MS [ ZAEHL [ 0 ] ] := MS
[ ZAEHL [ 0 ] ] OR 40
[X1B] 303: ELSE MS [ ZAEHL [ 0 ] ] := MS
[ ZAEHL [ 0 ] ] AND 215;
[AZ1] 304: IF FETT
[1CA] 305: THEN MS [ ZAEHL [ 0 ] ] := MS
[ ZAEHL [ 0 ] ] OR 8
[92B] 306: ELSE MS [ ZAEHL [ 0 ] ] := MS [
ZAEHL [ 0 ] ] AND 247;
[NK2] 307: IF HOCH THEN
[B21] 308: BEGIN
[5Z8] 309: MS [ ZAEHL [ 0 ] ] := MS [
ZAEHL [ 0 ] ] OR 1;
[502] 310: HT := TRUE;
[DD0] 311: END
[22B] 312: ELSE MS [ ZAEHL [ 0 ] ] := MS [
ZAEHL [ 0 ] ] AND 254;
[5L2] 313: IF TIEF THEN
[B21] 314: BEGIN
[8Z8] 315: MS [ ZAEHL [ 0 ] ] := MS [
ZAEHL [ 0 ] ] OR 2;
[MU1] 316: HT := TRUE
[DD0] 317: END
[Z1B] 318: ELSE MS [ ZAEHL [ 0 ] ] := MS [
ZAEHL [ 0 ] ] AND 253;
[BX3] 319: GEDRUCKT := FALSE;
[WK9] 320: DRUCKZEILE [ ZAEHL [ 0 ] ] :
= ORD ( ZEICHEN );
[7K4] 321: FOR ALAUF := 1 TO 12 DO
[B21] 322: BEGIN
[6F8] 323: IF OBENPUNKTE1 [ ORD (
ZEICHEN ), ALAUF ]
[ZBB] 324: OR ( UNTENPUNKTE [ ORD
( ZEICHEN ), ALAUF ] AND 128 )
[SE0] 325: <> 0
[EG6] 326: THEN ZAEHL [ 1 ] :=
ZAEHL [ 0 ];
[859] 327: IF ( OBENPUNKTE2 [ ORD
( ZEICHEN ), ALAUF ]
[KF9] 328: OR ( UNTENPUNKTE [ ORD
( ZEICHEN ), ALAUF ]
[687] 329: AND 8 )
[MG6] 330: THEN ZAEHL [ 2 ] :=
ZAEHL [ 0 ];
[ORA] 331: IF TRUNC ( UNTENPUNKTE
[ ORD ( ZEICHEN ), ALAUF ]
[ZT2] 332: AND 112
) / 16 <> 0
[UG6] 333: THEN ZAEHL [ 3 ] :=
ZAEHL [ 0 ];
[ENA] 334: IF UNTENPUNKTE [ ORD (
ZEICHEN ), ALAUF ] AND 7 <> 0
[VK6] 335: THEN ZAEHL [ 4 ] :=
ZAEHL [ 0 ];
[EB3] 336: END; ( ALAUF )
[EX2] 337: IF BREIT THEN
[B21] 338: BEGIN
[N64] 339: FOR ALAUF := 1 TO 4 DO
[4S6] 340: IF ZAEHL [ ALAUF ] = ZA
EHL [ 0 ]
[RD9] 341: THEN ZAEHL [ ALAUF ]
:= ZAEHL [ ALAUF ] + 1;
[T56] 342: ZAEHL [ 0 ] := ZAEHL [ 0 ]
+ 1;
[AH4] 343: MS [ ZAEHL [ 0 ] ] := 0;
[TMO] 344: END;
[BN3] 345: END; ( 1..6, ..... )
[2X2] 346: END; ( CASE )
[M85] 347: UNTIL ORD ( ZEICHEN ) = 26;
[5L0] 348: END.

```

Listing 4. Abgeänderte Version des Druckprogrammes für Drucker mit »9-Nadel-Grafik-Modus« (prn_9.pas)

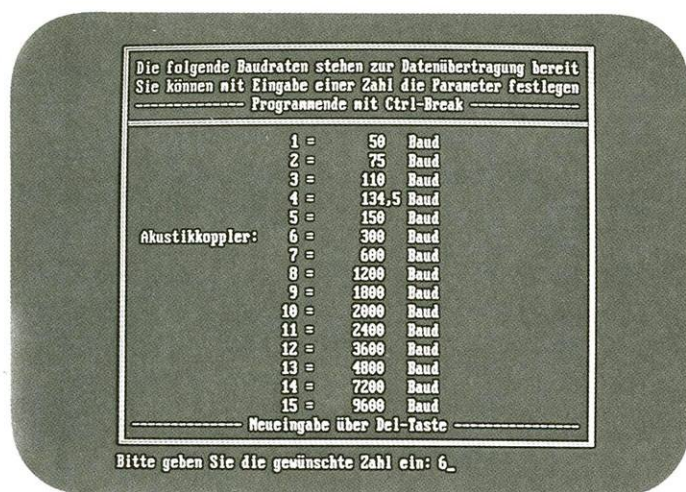


Bild 1. Der Menüpunkt zur Abfrage der Baudrate

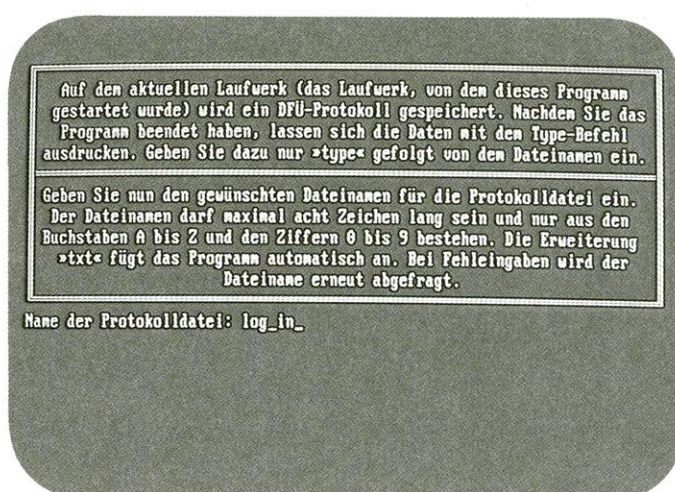


Bild 2. »STrans« schreibt alle empfangenen Daten in eine Protokolldatei

Datenfernübertragung mit »STrans«

Mit »STrans« steht Ihnen ab sofort ein einfaches Datenfernübertragungs-Programm für den täglichen Gebrauch zur Verfügung. Das Programm ist ausschließlich in Assembler geschrieben und erlaubt Datenübertragungsraten von 50 bis 9600 Baud.

»STrans« ist ein Programm, das die Kommunikation zwischen einem PC und einer Mailbox oder zwischen zwei PCs untereinander erlaubt. Sie können das Programm mit einem beliebigen Texteditor abtippen und von dem Makro-Assembler MASM übersetzen lassen. Wenn Sie die Unterschiede zwischen dem MASM und SAM II berücksichtigen, übersetzt Ihnen auch SAM II – nach den erforderlichen Änderungen selbstverständlich – das Programm problemlos.

Eine weitere, zeitsparende Alternative bietet Listing 2. Hier wurde jedes Byte des Programmcodes von STrans in eine zweistellige Hexadezimalzahl umgewandelt. Wenn Sie Listing 2, so wie es abgedruckt ist, aber ohne Prüfsummen und Zeilennummern eingeben, formt die Eingabehilfe »Scheck«, die ebenfalls in dieser Ausgabe abgedruckt ist, aus den Hexadezimalwerten eine Binärdatei, deren Endung »bin« Sie nur noch in »exe« umbenennen müssen, damit das Programm lauffähig ist.

Beachten Sie bei Listing 2, daß die Zeilen 3 bis 55 ausschließlich den Wert »00« enthalten (Prüfsumme »007«), und die Zeilen 369 bis 1043 nur aus der Hexadezimalzahl »20« (Prüfsumme »G37«) bestehen. Aus diesem Grund haben wir Ihnen den Abdruck dieser Zeilen erspart.

Gebrauchen Sie das Programm zum ersten Mal, so sind ein paar Menüabfragen zur Einstellung der Modem-Betriebsdaten zu beantworten. Der erste Menüpunkt fragt nach den Baudraten (Bild 1). Die Baudrate bezeichnet die Übertragungsgeschwindigkeit (Bit pro Sekunde), mit der Ihr Akustikkoppler oder Ihr Modem Daten übermittelt.

Arbeitet Ihr Akustikkoppler beispielsweise mit einer Übertragungsraten von 300 Baud, so müssen Sie im entsprechenden Menü die Zahl 6 eingeben. Bei einer Falschein-gabe erscheint das Menü erneut, und Sie können Ihre Eingabe korrigieren. Auch das Betätigen der Del-Taste führt zu einer Neueingabe.

Im zweiten Menüpunkt wird nach der Parität gefragt. Die Paritätsprüfung dient in der Datenfernübertragung dazu, um durch eine Zusatzinformation (das Paritätsbit) festzustellen, ob die Daten auf der Empfängerseite richtig emp-

fangen wurden. Weil die meisten Mailboxen jedoch auf eine Paritätsprüfung verzichten (das »N« in der Parameterangabe »8N1« steht für »No Parity«), empfiehlt es sich bei der Kommunikation mit solchen Boxen, auf ein Paritätsbit zu verzichten, und die Ziffer Null einzugeben.

Der dritte Menüpunkt verlangt Angaben über die Anzahl der Datenbits. Die Einstellung auf acht Datenbit bedeutet, daß der komplette ASCII-Zeichensatz, also auch Umlaute übertragen werden. Die Frage nach der Anzahl der Datenbits sollten Sie immer mit 8 beantworten.

Der vierte Menüpunkt fragt schließlich nach der Anzahl der Stoppbits. Gewöhnlich wird bei der Datenübertragung mit einem Stoppbit gearbeitet, so daß Sie die entsprechende Einstellung vornehmen müssen.

Ihre Standard-DFÜ-Parameter werden von STrans auf Wunsch dauerhaft gespeichert

Haben Sie alle Einstellungen vorgenommen, so können Sie auf Wunsch diese Konfiguration auf Datenträger speichern. STrans erzeugt zu diesem Zweck die Datei »dfu.kon«, die alle eingestellten Angaben enthält.

Anschließend benötigt das Programm noch einen Dateinamen für eine Protokolldatei, in der alle während der Datenübertragung empfangenen Zeichen gespeichert werden (Bild 2). Wenn Sie die Datenübertragung später mit der Funktionstaste [F10] oder durch Abschalten des Akustikkopplers beenden, wird auch die Protokolldatei abgeschlossen.

Möchten Sie sich die Protokolldatei anschauen, so geben Sie lediglich den MS-DOS-Befehl »type«, gefolgt von dem Namen der Protokolldatei an. Mit [Ctrl-S] halten Sie die Bildschirmausgabe an, und mit einer beliebigen Taste fahren Sie fort. Übrigens läßt sich auch in vielen Mailboxen die Bildschirmausgabe über die Tastenkombination [Ctrl-S] stoppen.

Wünschen Sie eine Ausgabe auf den Drucker, so fügen Sie die Zeichenkombination »>prn«, getrennt durch Leerzeichen, an, damit die Ausgabe des Dateiinhalts vom Bildschirm auf den Drucker umgeleitet wird.

Es kann nun auch sein, daß eine Datei mit dem von Ihnen eingegebenen Namen bereits bei einer vorhergehenden Datenübertragung erzeugt wurde. In diesem Fall fragt Sie STrans, ob das Überschreiben der alten Datei erwünscht ist. Wenn nicht, müssen Sie mit »N« für »Nein« antworten und wählen einen neuen Dateinamen.

(Hartmut Peyke/ma)


```

[Z94] 0: ;----- STrans 1.0
[HND] 1: ;----- Das DFÜ-Programm der DOS Internati
onal
[64B] 2: ;----- Copyright (c) by Hartmut Peyke
[B14] 3: ;----- Equates
[XU2] 4: BIOS EQU 10h
[FV2] 5: TAST EQU 16h
[KN2] 6: DOS EQU 21h
[90N] 7: RS EQU 3F8h ;Sende-Empfangspuffer und
Baudratenregister (Low-Byte)
[V10] 8: ;
[75M] 9: ; 3F9h Baudratenregister (Hi-By
te) und Unterbrechungsregister
[Y8E] 10: ; 3FAh Unterbrechungserkennungs
register
[22A] 11: ; 3FBh Leitungssteuerregister
[90A] 12: ; 3FCh Modemsteuerregister
[CVA] 13: ; 3FDh Leitungsstatusregister
[R2A] 14: ; 3FEh Modemstatusregister
[V10] 15: ;
[X78] 16: Stapel SEGMENT PARA STACK 'STACK'
[1X1] 17: DB 64 DUP(?)
[2E3] 18: Stapel ENDS
[V10] 19: ;
[RY5] 20: Daten SEGMENT PARA 'DATA'
[Y07] 21: ;----- Bildschirmmasken
[ATR] 22: ;----- In den folgenden Masken dürfen kei
ne Änderungen vorgenommen werden
[FC3] 23: Bd DB ' 50','$'
[CF3] 24: DB ' 75','$'
[KC2] 25: DB ' 110','$'
[557] 26: DB ' 134,5','$',' 150','$'
[677] 27: DB ' 300','$' ' 600','$',' 1200','$'
[V10] 28: ;
[UM9] 29: Bd_1 DB ' 2000','$',' 3600','$'
[YA7] 30: DB ' 2400','$' ' 4800','$',' 7200','$'
[V10] 31: ;
[RMD] 32: Parit_DAT DB ' keine','$',' ungerade','$'
' gerade','$'
[V10] 33: ;
[6F9] 34: D_Bit DB ' 5','$','$'
[PG6] 35: DB ' 6','$' ' 7','$' ' 8','$'
[OF4] 36: DB ' 7','$' ' 8','$'
[V10] 37: ;
[ZD4] 38: S_Bit DB '1','$','2','$'
[V10] 39: ;
[RFX] 40: Pruef_mask DB ' Von Ihnen eingest
ellt: '
[PS9] 41: DB ' Baud
[FFA] 42: DB ' Paritä
t
[GHA] 43: DB ' Datenb
it
[ORA] 44: DB ' Stoppb
it
[313] 45: DB '
[ZZ8] 46: DB 'Stimmen diese Angaben ? J/N
[2V0] 47: DB '$'
[92F] 48: Maske DB '
[1KR] 49: DB ' Die folgende Baudraten stehe
n zur Datenübertragung bereit '
[X7P] 50: DB ' Sie können mit Eingabe einer
Zahl die Parameter festlegen '
[5ZG] 51: DB ' Programmende
mit Ctrl-Break -----
[C8D] 52: DB '
[B8B] 53: DB ' 1 =
50 Baud '
[P9B] 54: DB ' 2 =
75 Baud '
[1AB] 55: DB ' 3 =
110 Baud '
[UDB] 56: DB ' 4 =
134,5 Baud '
[VAB] 57: DB ' 5 =
150 Baud '
[5RE] 58: DB ' Akustikkoppler: 6 =
300 Baud '
[WAB] 59: DB ' 7 =
600 Baud '

```

```

[2CB] 60: DB ' 8 =
1200 Baud '
[XCB] 61: DB ' 9 =
1800 Baud '
[7CB] 62: DB ' 10 =
2000 Baud '
[UCB] 63: DB ' 11 =
2400 Baud '
[8DB] 64: DB ' 12 =
3600 Baud '
[PDB] 65: DB ' 13 =
4800 Baud '
[ADB] 66: DB ' 14 =
7200 Baud '
[3EB] 67: DB ' 15 =
9600 Baud '
[NRG] 68: DB ' ----- Neueingabe üb
er Del-Taste -----
[MAD] 69: DB '
[K6E] 70: DB 'Bitte geben Sie die gewünschte
Zahl ein: '
[2V0] 71: DB '$'
[NFG] 72: Falsch DB ' Ungültige Zahl ! Neueingabe
über Del-Taste '
[2V0] 73: DB '$'
[6Z7] 74: Parity_mask DB '
[FWK] 75: DB ' Folgende Paritätsüberprüf
ungen stehen bereit '
[FYE] 76: DB ' ----- Programmende mit
Ctrl-Break -----
[4T3] 77: DB '
[Z1D] 78: DB ' 0 =
Keine Parität '
[W9D] 79: DB ' 1 =
Ungerade Parität '
[H5D] 80: DB ' 2 =
Gerade Parität '
[61F] 81: DB ' ----- Neueingabe über De
l-Taste -----
[KV3] 82: DB '
[K6E] 83: DB 'Bitte geben Sie die gewünsch
te Zahl ein: '
[2V0] 84: DB '$'
[M02] 85: Dabit_mask DB '
[MXF] 86: DB ' Die folgenden Datenbits st
ehen bereit '
[D8F] 87: DB ' ----- Programmende mit Ctrl
-Break ----
[UAZ] 88: DB '
[5UA] 89: DB ' 5 = 5 D
atenbits '
[CUA] 90: DB ' 6 = 6 D
atenbits '
[LUA] 91: DB ' 7 = 7 D
atenbits '
[UUA] 92: DB ' 8 = 8 D
atenbits '
[E3E] 93: DB ' ----- Neueingabe über Del-
Taste ----
[ADZ] 94: DB '
[K6E] 95: DB 'Bitte geben Sie die gewünsch
te Zahl ein: '
[2V0] 96: DB '$'
[SU2] 97: Stop_mask DB '
[G2G] 98: DB ' Die folgenden Stoppbits ste
hen bereit '
[D8F] 99: DB ' ----- Programmende mit Ctrl-
Break ----
[UAZ] 100: DB '
[LSA] 101: DB ' 1 = 1 Stoppbitt
'
[PSA] 102: DB ' 2 = 2 Stoppbitt
'
[E3E] 103: DB ' ----- Neueingabe über Del-T
aste ----
[ADZ] 104: DB '
[K6E] 105: DB 'Bitte geben Sie die gewünscht
e Zahl ein: '
[2V0] 106: DB '$'
[V10] 107: ;
[TS6] 108: DAT_Name DB 0Ah,0Ah,0Ah,0Dh
[UUN] 109: DB '
,0Ah,0Dh

```



```

[GHU] 110: DB ' Auf dem aktuellen Laufwerk
        (das Laufwerk, von dem dieses Programm
        ,0Ah,0Dh
[WST] 111: DB ' gestartet wurde) wird ein D
        FU-Protokoll gespeichert. Nachdem Sie das
        ,0Ah,0Dh
[W3T] 112: DB ' Programm beendet haben, la
        ssen sich die Daten mit dem Type-Befehl
        ,0Ah,0Dh
[F2V] 113: DB ' ausdrucken. Geben Sie dazu n
        ur »type« gefolgt von dem Dateinamen ein.
        ,0Ah,0Dh
[ASN] 114: DB '
        ,0Ah,0Dh
[FXT] 115: DB ' Geben Sie nun den gewünschte
        n Dateinamen für die Protokolldatei ein.
        ,0Ah,0Dh
[WGT] 116: DB ' Der Dateinamen darf maximal
        acht Zeichen lang sein und nur aus den
        ,0Ah,0Dh
[U7T] 117: DB ' Buchstaben A bis Z und den Z
        iffern 0 bis 9 bestehen. Die Erweiterung
        ,0Ah,0Dh
[GTT] 118: DB ' »txt« fügt das Programm au
        tomatisch an. Bei Fehleingaben wird der
        ,0Ah,0Dh
[4ZH] 119: DB ' Dateina
        me erneut abgefragt.
        ,0Ah,0Dh
[UUN] 120: DB '
        ,0Ah,0Dh
[2N8] 121: DB 'Name der Protokolldatei:
[2V0] 122: DB '$'
[V10] 123: ;
[A27] 124: Schreib DB 0Ah,0Ah,0Ah,0Dh
[UUN] 125: DB '
        ,0Ah,0Dh
[A2U] 126: DB ' Die Datei ist bereits vorha
        nden. Überschreiben trotzdem erwünscht?
        ,0Ah,0Dh
[UUN] 127: DB '
        ,0Ah,0Dh
[PC6] 128: DB 'Ihre Eingabe (J/N):
[2V0] 129: DB '$'
[V10] 130: ;
[5SB] 131: Param_Set DB 0Ah,0Ah,0Ah,0Dh ;Zeilenvorschu
        b
[6D6] 132: DB '
        ,0Ah,0Dh
[UWL] 133: DB ' Soll diese Konfiguration ab
        gespeichert werden?
        ,0Ah,0Dh
[7D6] 134: DB '
        ,0Ah,0Dh
[PC6] 135: DB 'Ihre Eingabe (J/N):
[2V0] 136: DB '$'
[V10] 137: ;
[NH6] 138: Konfig_mask DB 0Ah,0Ah,0Dh
[513] 139: DB '
        ,0Ah,0Dh
[T2H] 140: DB ' Es sind folgende Daten
        voreingestellt:
        ,0Ah,0Dh
[2V0] 141: DB '$'
[V10] 142: ;
[UCK] 143: Warten DB 0Ah,0Ah,0Ah,0Ah,0Ah,0Ah,0Ah,0Ah
        ,0Ah ;9mal Zeilenvorschub
[TNM] 144: DB '
        ,0Ah,0Dh
[WWC] 145: DB '
        ,0Ah,0Dh
[2PK] 146: DB ' Moment bitte ...
        ,0Ah,0Dh
[2V0] 147: DB '$'
[953] 148: ;Meldungen
[UVV] 149: Melde1 DB '1 Minute Wartezeit, innerhalb
        der sich die Gegenstelle mit Carrier melden
        muß.'
[2V0] 150: DB '$'
[000] 151:
[000] 152:
[8BR] 153: Melde2 DB 'Datenübertragung aktiv, Abbruc
        h über die Funktionstaste [F10]
[2V0] 154: DB '$'
[CVG] 155: Melde3 DB 'Datenübertragung inaktiv
        ,
[2V0] 156: DB '$'
[9F6] 157: ;Fehlertonerzeugung
[V93] 158: Beep DB 07h

```

```

[M21] 159: DB '$'
[NYA] 160: ;Speicher für Zeichenoperationen
[X25] 161: Eingabe DB 10 DUP (' ')
[X25] 162: Eingabe1 DB 10 DUP (' ')
[X25] 163: Eingabe2 DB 10 DUP (' ')
[X25] 164: Eingabe3 DB 10 DUP (' ')
[X25] 165: Eingabe4 DB 10 DUP (' ')
[W87] 166: ;----- DTA für Datei DFU.KON
[T18] 167: DTA_1 DB 515 DUP (' ');4mal 128+3 Byte
[PP7] 168: ;----- DTA für Datei DFU.TXT
[TSE] 169: DTA DB 10240 DUP (' ');Diskettenübert
        ragungsbereich
[MRE] 170: ;DTA in 128'er Blöcken (wird später in CX a
        ngegeben)
[CR2] 171: ;----- FCB's
[HT5] 172: FCB DB 0, ' TXT', 25 Dup(0)
[3FN] 173: ;Bei den FCB's für die Datei DFU.KON müssen
        diese zweimal vorhanden sein,
[K9N] 174: ;damit diese laufend überschrieben werden k
        ann (Satzzeiger auf Null)
[7C7] 175: FCB_Save DB 0,'DFU KON', 25 Dup(0)
[597] 176: FCB_Load DB 0,'DFU KON', 25 Dup(0)
[S8H] 177: ;FCB: 0=aktuelles Laufwerk, Name und Erweit
        erung der Datei
[V10] 178: ;
[Z22] 179: Daten ENDS
[V10] 180: ;
[DS3] 181: ;----- Makros
[944] 182: STRING_ANZ MACRO
[5YG] 183: MOV AH,9 ;Funktion 9 (Zeichenket
        te aus Speicher anzeigen)
[8S9] 184: INT DOS ;Funktionsaufruf 21hex
[1R0] 185: ENDM
[V10] 186: ;
[X04] 187: ZEICH_ANZ MACRO
[Y4G] 188: MOV AH,1 ;Funktion 1 (auf Zeichen
        von Tastatur warten und
[T8C] 189: INT DOS ;aufrufen) Funktionsaufr
        uf 21hex
[1R0] 190: ENDM
[V10] 191: ;
[WP3] 192: CURS_SET MACRO
[MOF] 193: MOV AH,2 ;Cursor positionieren, D
        L=Spalte,DH=Zeile
[FE9] 194: INT BIOS ;Funktionsaufruf 10hex
[1R0] 195: ENDM
[V10] 196: ;
[KN3] 197: CURS_LES MACRO
[NZE] 198: MOV AH,3 ;Cursorposition lesen, D
        L=Spalte,DH=Zeile
[AW1] 199: INT BIOS
[1R0] 200: ENDM
[V10] 201: ;
[ZC3] 202: CURS_24 MACRO
[69A] 203: MOV DL,0 ;Setze Cursor auf Spalte
        0
[87A] 204: MOV DH,24 ;Setze Cursor auf Zeile
        24
[Z62] 205: CURS_SET
[1R0] 206: ENDM
[V10] 207: ;
[783] 208: CURS_0 MACRO
[FDD] 209: MOV DX,0 ;Setze Cursor auf Spalte
        und Zeile 0
[Z62] 210: CURS_SET
[1R0] 211: ENDM
[8A4] 212: ;----- Makroende
[V10] 213: ;
[PH5] 214: Code SEGMENT PARA 'CODE'
[NP4] 215: Programm PROC FAR
[WB9] 216: ASSUME CS:Code, DS:Daten, SS:Stape
        l
[V10] 217: ;
[GR7] 218: STI ;Freigabe Interrupt
[WE9] 219: PUSH DS ;Rückkehradresse zu DOS
[H67] 220: MOV AX,0 ;auf Stack ablegen
[1X1] 221: PUSH AX
[DM8] 222: MOV AX, Daten ;Datenbereich in
[2M8] 223: MOV DS, AX ;DS initialisieren
[F7G] 224: ;Zunächst wird die aktuelle Konfiguration g
        eladen
[T9K] 225: ;und gefragt, ob diese bestehen unverändert
        bleiben soll
[RW9] 226: CALL Bild_Cls ;Bildschirm löschen
[CLA] 227: CURS_0 ;Cursor auf Zeile/S
        palte 0
[3PF] 228: CALL Load_Konfig ;hole Konfigurati
        on von Diskette
[9SA] 229: CMP AL,255 ;Datei ist nicht vo
        rhanden
[429] 230: JE Baud_Set ;dann erzeuge diese
[CAZ] 231: Frage_LoKon: MOV AH,0 ;Taste einlesen
[G1Y] 232: INT TAST
[4V4] 233: CMP AL,'j' ;klein ja

```



```

[GOC] 234:      JE OK      ;dann Konfiguration
[1D4] 235:      CMP AL,'J' ;groß JA
[GOC] 236:      JE OK      ;dann Konfiguration
[516] 237:      CMP AL,'n' ;klein nein
[PGC] 238:      JE Baud_set ;gehe zum Paramet
[RE5] 239:      CMP AL,'N' ;groß NEIN
[PGC] 240:      JE Baud_set ;gehe zum Paramet
[35E] 241:      MOV DX,Offset Beep ;Fehlerton
[WDH] 242:      STRING ANZ ;Fehlerton ausgeb
[DDD] 243:      JMP Frage_LoKon ;weitere Ze
[10] 244:      ;
[HUD] 245:      OK:      JMP Warte ;Sprung, da Konfigurati
[CTL] 246:      ;Nachfolgend soll der Baudratengenerator pr
[TKH] 247:      ;Dazu muß Bit 7 im Leitungssteuerregister g
[TA] 248:      Baud_Set: MOV DX,RS+3 ;Leitungssteuerregist
[3V] 249:      MOV AL,80h ;(80hex)
[3D4] 250:      OUT DX,AL ;setzen
[3V] 251:      ;Maske und Eingabe der Baudrate
[CTB] 252:      Mask1: CALL Bild_Cls ;Bildschirm löschen
[3V] 253:      CURS 0 ;Cursor auf Zeile
[14B] 254:      MOV DX,Offset Maske ;gebe Eingabem
[3V] 255:      STRING ANZ ;aus
[10] 256:      ;
[3V] 257:      Ein:      MOV DI,Offset Eingabe ;hole Zahl
[3V] 258:      MOV BL,0 ;Zähler
[3V] 259:      Schl:      MOV AH,1 ;zeige Zahl
[3V] 260:      INT DOS ;an
[3V] 261:      CMP AL,08h ;Del-Taste zum Löschen
[3V] 262:      gedrückt ?
[3V] 263:      JE Mask1 ;Ja, dann Neueingabe
[3V] 264:      CMP AL,39h ;Zeichen größer als 9
[3V] 265:      JA Springe ;= Fehler
[3V] 266:      CMP AL,0Dh ;Enter-Taste zur Bestät
[3V] 267:      igung gedrückt ?
[3V] 268:      JE Eine_Z
[3V] 269:      CMP AL,30h ;Zeichen kleiner als Nu
[3V] 270:      ll
[3V] 271:      JB Springe ;prüfe auf Fehler
[3V] 272:      Sich:      MOV [DI],AL ;Zeichen in DI sichern
[3V] 273:      INC DI ;DI erhöhen
[3V] 274:      INC BL ;Zähler erhöhen
[3V] 275:      CMP BL,3 ;mehr als zwei Zahlen
[3V] 276:      JE Mask1 ;dann Neueingabe
[3V] 277:      CURS LES ;lese Cursorposition
[3V] 278:      CMP DL,44 ;3 Zahlen eingeben (hie
[3V] 279:      r sind bereits 41 Zeichen)
[3V] 280:      JAE Springe ;= Falsch
[3V] 281:      JMP Schl
[3V] 282:      ;
[3V] 283:      Springe: CALL False ;springe zum Unterproc
[3V] 284:      JMP Ein ;danach alles von neuem
[3V] 285:      Eine_Z:      CMP BL,0 ;BL=0, noch keine Zahl e
[3V] 286:      ingegeben
[3V] 287:      JE Mask1 ;aber Enter-Taste betäti
[3V] 288:      gen=Neu
[3V] 289:      CMP BL,2 ;etwa zwei Zahlen
[3V] 290:      JE Summ ;dann addiere
[3V] 291:      MOV SI,Offset Eingabe ;nur eine Za
[3V] 292:      hl, dann hole Zahl
[3V] 293:      MOV AL,[SI] ;übergebe in AL
[3V] 294:      SUB AL,30h ;nur eine Zahl, also m
[3V] 295:      inus 30hex
[3V] 296:      INC SI ;SI erhöhen
[3V] 297:      MOV AH,0Dh ;AH mit [Enter] laden
[3V] 298:      und in SI sichern
[3V] 299:      MOV [SI],AH ;um bei späteren Opera
[3V] 300:      tionen ein Ende-
[3V] 301:      ;Zeichen zu haben
[3V] 302:      ;
[3V] 303:      Aus:      MOV DX,RS ;Baudratenregister (Low)
[3V] 304:      vorbereiten
[3V] 305:      ;
[3V] 306:      CMP AL,0
[3V] 307:      JE Mask1
[3V] 308:      CMP AL,1 ;Eins eingegeben ?
[3V] 309:      JE 50 ;Ja, dann setze 50 Baud
[3V] 310:      CMP AL,2 ;Zwei eingegeben ?
[3V] 311:      JE 75 ;usw.
[3V] 312:      CMP AL,3
[3V] 313:      JE 110
[3V] 314:      CMP AL,4

```

```

[3K1] 305:      JE 134
[EM1] 306:      CMP AL,5
[1H1] 307:      JE 150
[EM1] 308:      CMP AL,6
[2G1] 309:      JE 300
[EM1] 310:      CMP AL,7
[EH1] 311:      JE 600
[EM1] 312:      CMP AL,8
[1H1] 313:      JE 1200
[EM1] 314:      CMP AL,9
[5K1] 315:      JE 1800
[592] 316:      JMP Ein
[PLA] 317:      ;----- Tabelle für Baudraten bis 300 Baud
[2C3] 318:      _50:      MOV AL,00h
[ER1] 319:      JMP 50_1
[NC3] 320:      _75:      MOV AL,00h
[XS1] 321:      JMP 75_1
[6F3] 322:      _110:     MOV AL,17h
[DV1] 323:      JMP 110_1
[4G3] 324:      _134:     MOV AL,59h
[UN1] 325:      JMP 135
[B53] 326:      _150:     MOV AL,00
[9W1] 327:      JMP 150_1
[OF3] 328:      _300:     MOV AL,80h
[KV1] 329:      JMP 300_1
[KW8] 330:      ;----- zwei Zahlen eingegeben
[W5F] 331:      Springe3: CALL False ;springe zur Unterproz
[3V] 332:      edur
[3V] 333:      Summ:      MOV SI,Offset Eingabe
[3V] 334:      MOV AH,[SI]
[3V] 335:      SUB AH,30h
[3V] 336:      CMP AH,2 ;erste Zahl größer a
[3V] 337:      ls Zwei
[3V] 338:      JAE Springe3 ;=falsche Eingabe
[3V] 339:      Zweite:      INC SI ;zeige auf zweite Za
[3V] 340:      hl
[3V] 341:      MOV AL,[SI] ;hole zweite Zahl
[3V] 342:      SUB AL,30h ;minus 30hex und erh
[3V] 343:      alte Originalwert
[3V] 344:      CMP AL,5 ;zweite Zahl etwa gr
[3V] 345:      ößer als 5
[3V] 346:      JA Springe3 ;das wäre mindestens
[3V] 347:      16 und falsch
[3V] 348:      ADD AL,AH ;addiere jetzt Zahl
[3V] 349:      1+2
[3V] 350:      CMP AL,7 ;vergleiche auf Summ
[3V] 351:      =7, wenn größer,
[3V] 352:      JAE Springe3 ;oder gleich dann Fe
[3V] 353:      hltermeldung
[3V] 354:      INC SI ;SI erhöhen
[3V] 355:      MOV AH,0Dh ;AH mit [Enter] lade
[3V] 356:      n und in SI sichern
[3V] 357:      MOV [SI],AH ;um bei späteren Ope
[3V] 358:      rationen ein Ende
[3V] 359:      ;Zeichen zu haben
[3V] 360:      ;
[3V] 361:      Aus:      MOV DX,RS ;Baudratenregister (Low)
[3V] 362:      vorbereiten
[3V] 363:      ;
[3V] 364:      CMP AL,1 ;zwei Zahlen z.B.10=1+0=
[3V] 365:      1
[3V] 366:      JE 2000 ;wenn Eins, dann setze 2
[3V] 367:      000 Baud
[3V] 368:      CMP AL,2 ;usw.
[3V] 369:      JE 2400
[3V] 370:      CMP AL,3
[3V] 371:      JE 3600
[3V] 372:      CMP AL,4
[3V] 373:      JE 4800
[3V] 374:      CMP AL,5
[3V] 375:      JE 7200
[3V] 376:      CMP AL,6
[3V] 377:      JE 9600
[3V] 378:      JMP Mask1
[3V] 379:      ;----- Tabelle für Baudraten ab 600 Baud
[3V] 380:      MOV AL,0C0h ;AL für 600 Baud vorb
[3V] 381:      ereiten
[3V] 382:      JMP Aus1 ;und einstellen
[3V] 383:      _1200:      MOV AL,60h
[3V] 384:      JMP Aus1
[3V] 385:      _1800:      MOV AL,40h
[3V] 386:      JMP Aus1
[3V] 387:      _2000:      MOV AL,3Ah
[3V] 388:      JMP Aus1
[3V] 389:      _2400:      MOV AL,30h
[3V] 390:      JMP Aus1
[3V] 391:      _3600:      MOV AL,20h
[3V] 392:      JMP Aus1
[3V] 393:      _4800:      MOV AL,18h
[3V] 394:      JMP Aus1
[3V] 395:      _7200:      MOV AL,10h
[3V] 396:      JMP Aus1

```



```

[CS3] 383: _9600: MOV AL,0Ch
[WB2] 384: JMP Aus1
[V10] 385: ;
[2RC] 386: Aus1: OUT DX,AL ;gebe vorbesetzten We
rt aus
[UYE] 387: MOV DX,RS+1 ;Baudratenregister-Hi
gh (alle 600 bis 9600 Baud)
[DS3] 388: MOV AL,00h ;=00hex
[3D4] 389: OUT DX,AL ;setzen
[EN2] 390: JMP Parity
[YW4] 391: ;----- 50 bis 300 Baud
[POE] 392: _50_1: OUT DX,AL ;bei Baudraten von 50
bis 300 Baud
[9LF] 393: MOV DX,RS+1 ;erhält das Baudraten
register-High einen
[R2H] 394: MOV AL,09h ;von Baudrate zu Baud
rate unterschiedlichen
[1NF] 395: OUT DX,AL ;Wert, im Gegensatz z
u Baudraten ab 600 Baud
[CRF] 396: JMP Parity ;wo im Baudratenregis
ter-High konstant
[8VA] 397: _75_1: OUT DX,AL ;00hex eingegeben wir
d
[7Z8] 398: MOV DX,RS+1 ;deshalb diese Tabell
e
[MX1] 399: MOV AL,06h
[TU1] 400: OUT DX,AL
[EN2] 401: JMP Parity
[YS3] 402: _110_1: OUT DX,AL
[122] 403: MOV DX,RS+1
[KX1] 404: MOV AL,04h
[TU1] 405: OUT DX,AL
[EN2] 406: JMP Parity
[3B3] 407: _135: OUT DX,AL
[122] 408: MOV DX,RS+1
[HX1] 409: MOV AL,03h
[TU1] 410: OUT DX,AL
[EN2] 411: JMP Parity
[AT3] 412: _150_1: OUT DX,AL
[122] 413: MOV DX,RS+1
[HX1] 414: MOV AL,03h
[TU1] 415: OUT DX,AL
[EN2] 416: JMP Parity
[ZS3] 417: _300_1: OUT DX,AL
[122] 418: MOV DX,RS+1
[FX1] 419: MOV AL,01h
[TU1] 420: OUT DX,AL
[EN2] 421: JMP Parity
[B14] 422: ;----- Parität
[82N] 423: ;Bit 7 im Baudratengenerator muß wieder auf
Null gesetzt werden
[D7G] 424: Parity: MOV DX,RS+3 ;Leitungssteuerregi
ster Bit 7 = 0
[BW2] 425: MOV AL,00h ;
[BF4] 426: OUT DX,AL ;setzen
[V10] 427: ;
[AX9] 428: CALL Bild_Cls ;Bildschirm löschen
[PZ7] 429: CURS 0 ;Cursor auf Zeile
[T7D] 430: MOV DX,Offset Parity_mask ;gebe E
ingabemaske
[SS3] 431: STRING_ANZ ;aus
[V10] 432: ;
[VKB] 433: Ein1: MOV DI,Offset Eingabe1 ;hole Zahl
[LN1] 434: MOV BL,0
[GA7] 435: Schl1: MOV AH,1 ;zeige Zahl
[Z33] 436: INT DOS ;an
[66E] 437: CMP AL,08h ;Del-Taste zur Best
ätigung gedrückt ?
[W7H] 438: JE Parity ;wenn Del-Taste abe
r noch kein Zeichen, löschen
[S9A] 439: CMP AL,32h ;Zeichen größer als
Zwei ?
[H06] 440: JA Springel ;=Fehler
[6Z5] 441: CMP AL,0Dh ;Enter-Taste ?
[PG1] 442: JE BLO
[MBA] 443: CMP AL,30h ;Zeichen kleiner al
s Null ?
[XD5] 444: JB Springel ;Fehler
[HCC] 445: Sicher: MOV [DI],AL ;Zeichen in DI sich
ern
[S91] 446: INC DI
[0A1] 447: INC BL
[CF9] 448: CURS_LES ;Cursorposition les
en
[GP1] 449: CMP DL,43
[NL2] 450: JAE Parity
[EA2] 451: JMP Schl1
[C2B] 452: Springel: CALL False ;Fehlermeldung
[W4B] 453: JMP Ein1 ;noch einmal Zeiche
n holen
[PXD] 454: BLO: CMP BL,0 ;Enter-Taste aber n
och kein Zeichen
[CEE] 455: JE Parity ;dann alles noch ei
nmal (Zähler BL=Null)

```

```

[8T7] 456: ;----- Tabelle zur Parität
[PYF] 457: Tabloe: MOV SI,Offset Eingabe1 ;sonst Zei
chen in
[AD6] 458: MOV AL,[SI] ;AL übergeben
[4W1] 459: SUB AL,30h
[8BE] 460: MOV DX,RS+3 ;Leitungssteuerregi
ster vorbelegen
[EM1] 461: CMP AL,0
[HE2] 462: JE Keine
[EM1] 463: CMP AL,1
[1G2] 464: JE Gerade
[EM1] 465: CMP AL,2
[L13] 466: JE Ungerade
[EN2] 467: JMP Parity
[V10] 468: ;
[595] 469: Keine: MOV AL,00000000b
[MC3] 470: JMP Pari_aus
[OP5] 471: Gerade: MOV AL,00010000b
[MC3] 472: JMP Pari_aus
[S16] 473: Ungerade: MOV AL,00000000b
[MC3] 474: JMP Pari_aus
[V10] 475: ;
[E8A] 476: Pari_aus: OUT DX,AL ;Parität raus
[5G4] 477: ;----- Datenbits
[23C] 478: BITS: CALL Bild_Cls ;Bildschirm löschen
[618] 479: CURS 0 ;Cursor auf Zeile 0
[WEC] 480: MOV DX,Offset Dabit_mask ;gebe Ei
ngabemaske
[SS3] 481: STRING_ANZ ;aus
[V10] 482: ;
[ZKB] 483: Ein2: MOV DI,Offset Eingabe2 ;hole Zahl
[LN1] 484: MOV BL,0
[NA7] 485: Schl2: MOV AH,1 ;zeige Zahl
[Z33] 486: INT DOS ;an
[66E] 487: CMP AL,08h ;Del-Taste zur Best
ätigung gedrückt ?
[57H] 488: JE Bits ;wenn Del-Taste abe
r noch kein Zeichen, löschen
[DD8] 489: CMP AL,38h ;Zeichen größer als
8
[M06] 490: JA Springe2 ;=Fehler
[2X1] 491: CMP AL,0Dh
[4F2] 492: JE Tabell
[H88] 493: CMP AL,35h ;Zeichen kleiner al
s 5
[OE5] 494: JB Springe2 ;Fehler
[TAC] 495: Sicher2: MOV [DI],AL ;Zeichen in DI sich
ern
[S91] 496: INC DI
[0A1] 497: INC BL
[CF9] 498: CURS_LES ;Cursorposition les
en
[NZ7] 499: CMP DL,43 ;Spalte 43 erreicht
[EPD] 500: JAE Bits ;größer oder gleich
, dann Sprung
[FA2] 501: JMP Schl2
[C2B] 502: Springe2: CALL False ;Fehlermeldung
[W4B] 503: JMP Ein2 ;noch einmal Zeiche
n holen
[637] 504: ;----- Tabelle Datenbits
[H84] 505: Tabell: CMP BL,0
[6H2] 506: JE Bits
[9WA] 507: MOV SI,Offset Eingabe2 ;sonst Zei
chen in
[XC7] 508: MOV AL,[SI] ;AL überge
ben
[412] 509: Sub AL,30h
[8BE] 510: MOV DX,RS+3 ;Leitungssteuerregi
ster vorbelegen
[EM1] 511: CMP AL,5
[NF2] 512: JE Fuenf
[EM1] 513: CMP AL,6
[3D2] 514: JE Sechs
[EM1] 515: CMP AL,7
[7G2] 516: JE Sieben
[EU1] 517: CMP AL,8
[3E2] 518: JE Acht
[0A2] 519: JMP Bits
[T95] 520: Fuenf: MOV AH,00000000b
[DM2] 521: JMP Rausda
[AA5] 522: Sechs: MOV AH,00000001b
[DM2] 523: JMP Rausda
[7R5] 524: Sieben: MOV AH,00000010b
[DM2] 525: JMP Rausda
[UX4] 526: Acht: MOV AH,00000011b
[DM2] 527: JMP Rausda
[V10] 528: ;
[YNG] 529: Rausda: IN AL,DX ;bereits gesetzten
Wert einlesen
[ULF] 530: OR AL,AH ;ODER verknüpfen (a
lle Bits bleiben gesetzt)
[K66] 531: OUT DX,AL ;und ausgeben

```



```

[NB5] 532: ;----- Stoppbits
[STC] 533: Stopp: CALL Bild_Cls ;Bildschirm löschen
[618] 534: CURS_0 ;Cursor auf Zeile 0
[R5C] 535: MOV DX,Offset Stop_mask ;gebe Ein
gabemaske
[855] 536: STRING_ANZ ;aus
[V10] 537: ;
[T9C] 538: Ein4: MOV DI,Offset Eingabe3 ;hole Zah
1
[LN1] 539: MOV BL,0
[ZA7] 540: Schl4: MOV AH,1 ;zeige Zahl
[Z33] 541: INT DOS ;an
[66E] 542: CMP AL,08h ;Del-Taste zur Best
ätigung gedrückt ?
[D9H] 543: JE Stopp ;wenn Del-Taste abe
r noch kein Zeichen, löschen
[S9A] 544: CMP AL,32h ;Zeichen größer als
Zwei ?
[U06] 545: JA Springe4 ;=Fehler
[0FE] 546: CMP AL,0Dh ;Enter-Taste zur Be
stätigung gedrückt ?
[G3L] 547: JE Tabloel ;wenn Enter-Taste a
ber noch kein Zeichen, löschen
[LEA] 548: CMP AL,31h ;Zeichen kleiner al
s Eins
[6E5] 549: JB Springe4 ;Fehler
[YNC] 550: Sichere: MOV [DI],AL ;Zeichen in DI sich
ern
[S91] 551: INC DI
[0A1] 552: INC BL
[CF9] 553: CURS_LES ;Cursorposition les
en
[GP1] 554: CMP DL,43
[BF2] 555: JAE Stopp
[HA2] 556: JMP Schl4
[C2B] 557: Springe4: CALL False ;Fehlermeldung
[W4B] 558: JMP Ein4 ;noch einmal Zeiche
n holen
[XX6] 559: ;----- Tabelle Stoppbit
[L64] 560: Tabloel: CMP BL,0
[EL2] 561: JE Stopp
[FWA] 562: MOV SI,Offset Eingabe3 ;sonst Zei
chen in
[XC7] 563: MOV AL,[SI] ;AL überge
ben
[412] 564: Sub AL,30h
[8BE] 565: MOV DX,RS+3 ;Leitungssteuerregi
ster vorbelegen
[EM1] 566: CMP AL,1
[UC2] 567: JE Bit_1
[EM1] 568: CMP AL,2
[UC2] 569: JE Bit_2
[4H2] 570: JMP Stopp
[V10] 571: ;
[505] 572: Bit_1: MOV AH,00000000b
[0G3] 573: JMP Stop_aus
[H05] 574: Bit_2: MOV AH,00000100b
[0G3] 575: JMP Stop_aus
[V10] 576: ;
[TAF] 577: Stop_aus: IN AL,DX ;bereits gesetzten Wert
einlesen
[YKF] 578: OR AL,AH ;ODER verknüpfen (Alle
Bits bleiben gesetzt)
[AR6] 579: OUT DX,AL ;und ausgeben
[V10] 580: ;
[0PE] 581: ;letzte Abfrage, ob alles richtig (Baudrate
n)
[MFD] 582: The_last: CALL Bild_Cls ;Bildschirm löschen
[UE8] 583: CURS_0 ;Cursor Zeile, Spal
te 0
[62C] 584: MOV DX,Offset Pruef_mask ;gebe Fr
agemaske
[SS3] 585: STRING_ANZ ;aus
[V10] 586: ;
[4LG] 587: MOV SI,Offset Eingabe ;Zeiger auf
eingegebene Zahl Baudrate
[ZP6] 588: INC SI ;Zeiger erhöhen
[0D7] 589: MOV AL,[SI] ;in AL übergeben
[LFG] 590: CMP AL,0Dh ;schon Enter erreich
t? (ein Zeichen eingegeben)
[Y99] 591: JE Weniger ;dann bearbeite dies
e
[S2E] 592: INC SI ;wurden zwei Zahlen
eingegeben? (1 bis 15)
[L67] 593: MOV AL,[SI] ;dann übergeben
[5FC] 594: CMP AL,0Dh ;auf Ende-Zeichen pr
üfen (Enter)
[GCH] 595: JE Mehr ;und bearbeiten (nur
hier zwei Zeichen möglich)
[V10] 596: ;
[CXN] 597: Weniger: MOV SI,Offset Eingabe ;es wurde nu
r eine Zahl eingegeben (1 bis 9)
[2XA] 598: MOV AL,[SI] ;nochmals in AL über
geben

```

```

[825] 599: SUB AL,30h ;minus 30hex
[554] 600: CMP AL,1 ;wenn>=1
[ZTB] 601: JAE Bis_1800 ;springe Anzeigen bi
s 1800 Baud
[D64] 602: CMP AL,9 ;wenn<=9
[1UB] 603: JBE Bis_1800 ;springe Anzeigen bi
s 1800 Baud
[5R9] 604: JMP The_last ;nur bei Falsch-Wert
en
[V10] 605: ;
[HMB] 606: Bis_1800: CURS_LES ;Cursorposition lese
n
[LH6] 607: PUSH DX ;sichere diese
[K4A] 608: MOV DL,12 ;setze Cursor auf Sp
alte 12
[9Z5] 609: MOV DH,1 ;und Zeile 1
[VP6] 610: CURS_SET ;setze Cursor
[DRA] 611: MOV DX,Offset Bd ;hole Text (Baud
raten)
[B18] 612: CMP AL,1 ;war es eine Eins ?
[GUB] 613: JE Zeige ;dann zeige 50 Baud
sofort an
[EXE] 614: MOV BL,11 ;größer als eins, da
nn lade BL mit 11 für
[F3D] 615: MUL BL ;Multiplikation, Erg
ebnis dieser in AL
[VDD] 616: Vergl: CMP AL,1 ;AL kann jetzt nicht ein
s sein
[YPE] 617: JE Zeige ;erst nach dem folgenden
erniedrigt
[VP9] 618: INC DX ;Zeiger (Offset erhöhen)
[KSD] 619: DEC AL ;AL sooft erniedrigen bi
s Zeiger (DX) den
[51H] 620: JMP Vergl ;Wert hat, um richtige B
audrate anzuzeigen
[UYD] 621: Zeige: STRING_ANZ ;zeige diesen Wert jetz
t an
[OM5] 622: POP DX ;Cursor auf
[P24] 623: INC DL ;alte
[189] 624: CURS_SET ;Position+1 (INC DL) set
zen
[RDF] 625: JMP The_Last1 ;gehe zur Abfrage, o
b Parität richtig
[V10] 626: ;
[YNH] 627: Mehr: MOV SI,Offset Eingabe ;zwei Zahlen
wurden eingegeben
[76A] 628: MOV AH,[SI] ;übergebe erste Zahl
in AH
[G15] 629: SUB AH,30h ;minus 30hex
[ZP6] 630: INC SI ;Zeiger erhöhen
[CEA] 631: MOV AL,[SI] ;übergebe zweite Zah
l in AL
[825] 632: SUB AL,30h ;minus 30hex
[808] 633: ADD AL,AH ;addiere beide Zahle
n
[V10] 634: ;
[7BF] 635: ;auch hier kann Ergebnis nur 1 bis 6 sein (
10=1+0=1,15=1+5=6)
[V10] 636: ;
[9D4] 637: CMP AL,1 ;wenn>=1
[5YA] 638: JAE ab_2000 ;springe Anzeigen bis
1800 Baud
[TD4] 639: CMP AL,6 ;wenn<=9
[7YA] 640: JBE ab_2000 ;springe Anzeigen bis
1800 Baud
[2C3] 641: JMP The_last
[VX0] 642: ;
[RTB] 643: Ab_2000: CURS_LES ;Cursorposition lesen
[347] 644: PUSH DX ;sichere diese
[PNA] 645: MOV DL,12 ;setze Cursor auf Spalte
12
[X55] 646: MOV DH,1 ;und Zeile 1
[Z62] 647: CURS_SET
[MAB] 648: MOV DX,Offset Bd_1 ;hole Text (Bau
draten)
[F17] 649: CMP AL,1 ;war es eine Eins ?
[DH9] 650: JE Zeige_1 ;dann zeige 2000 Baud
an
[5TD] 651: MOV BL,11 ;größer als 1, dann lade
BL mit 11 für
[EMB] 652: MUL BL ;Multiplikation Ergebnis
in AL
[GRD] 653: Vergl_1: CMP AL,1 ;AL kann jetzt nicht Ein
s sein
[7UE] 654: JE Zeige_1 ;erst nach dem folgende
n erniedrigt
[VP9] 655: INC DX ;Zeiger (Offset erhöhen)
[KSD] 656: DEC AL ;AL sooft erniedrigen bi
s Zeiger (DX) den
[BWG] 657: JMP vergl_1 ;Wert hat um richtige
Baudrate anzuzeigen
[F9E] 658: Zeige_1: STRING_ANZ ;zeige diesen Wert jetz
t an

```



```
[OM5] 659:      POP DX      ;Cursor auf
[P24] 660:      INC DL      ;alte
[189] 661:      CURS_SET    ;Position+1 (INC DL) set
zen
[UU5] 662: ;----- Baudrate Ende
[LZD] 663: ;letzte Abfrage, ob alles richtig (Parität)
[3NP] 664: The_Last1: MOV SI,Offset Eingabel ;es wurde
nur eine Zahl eingegeben (0 bis 2)
[PD7] 665:      MOV AL,[SI] ;in AL übergeben
[1G5] 666:      SUB AL,30h ;minus 30hex
[T74] 667:      CMP AL,0 ;wenn=0
[VS7] 668:      JAE Parityx ;springe Parity
[284] 669:      CMP AL,2 ;wenn<=3
[XS7] 670:      JBE Parityx ;springe Parity
[UT9] 671:      JMP The_last1 ;nur bei Falsch-We
rten
[KNC] 672: Parityx: CURS_LES ;Cursorposition lesen
[LH6] 673:      PUSH DX ;sichere diese
[X4A] 674:      MOV DL,12 ;setze Cursor auf Spa
lte 12
[AZ5] 675:      MOV DH,2 ;und Zeile 2
[VP6] 676:      CURS_SET ;setze Cursor
[R8C] 677:      MOV DX,Offset Parit_Dat ;hole Text
(Parität)
[2W7] 678:      CMP AL,0 ;war es eine Null ?
[UBE] 679:      JE Zeigel2 ;dann zeige sofort ke
ine Parität an
[PVE] 680:      MOV BL,10 ;größer als Null, dan
n lade BL mit 10 für
[F3D] 681:      MUL BL ;Multiplikation, Erge
bnis dieser in AL
[CLD] 682: Vergl12: CMP AL,0 ;AL kann jetzt nicht
Null sein
[29E] 683:      JE Zeigel2 ;erst nach dem folge
nden erniedrigt
[GW8] 684:      INC DX ;Zeiger (Offset erhöh
en)
[F2E] 685:      DEC AL ;AL sooft erniedrigen
bis Zeiger (DX) den
[7FG] 686:      JMP vergl12 ;Wert hat, um richtig
e Parität anzuzeigen
[N2B] 687: Zeigel2: STRING_ANZ ;zeige diesen Wert an
[E17] 688:      POP DX ;Cursor auf alte
[AN7] 689:      CURS_SET ;Position setzen
[3M5] 690: ;----- Parität Ende
[08E] 691: ;letzte Abfrage, ob alles richtig (Datenbit
)
[NKN] 692: The_Last2: MOV SI,Offset Eingabe2 ;es wurde
nur eine Zahl eingegeben (5-8)
[PD7] 693:      MOV AL,[SI] ;in AL übergeben
[1G5] 694:      SUB AL,30h ;minus 30hex
[K84] 695:      CMP AL,5 ;wenn=5
[SH8] 696:      JAE Dabit_x ;springe Datenbit
[V84] 697:      CMP AL,8 ;wenn<=8
[UH8] 698:      JBE Dabit_x ;springe Datenbit
[ZT9] 699:      JMP The_last2 ;nur bei Falsch-We
rten
[34D] 700: Dabit_x: CURS_LES ;Cursorposition lesen
[347] 701:      PUSH DX ;sichere diese
[PNA] 702:      MOV DL,12 ;setze Cursor auf Spalte
12
[965] 703:      MOV DH,3 ;und Zeile 3
[7W7] 704:      CURS_SET ;setze Cursor
[9HB] 705:      MOV DX,Offset D_Bit ;hole Text (Da
tenbit)
[M36] 706:      CMP AL,5 ;war es eine 5 ?
[6EE] 707:      JE Zeigel23 ;dann zeige sofort ke
ine Parität an
[1UD] 708:      MOV BL,11 ;größer als 5, dann lade
BL mit 11 für
[V0E] 709:      MUL BL ;Multiplikation, Ergebni
s dieser in AL
[4CD] 710: Vergl123: CMP AL,0 ;AL kann jetzt nicht Nul
l sein
[CXE] 711:      JE Zeigel23 ;erst nach dem folgen
den erniedrigt
[CV9] 712:      INC DX ;Zeiger (Offset erhöhen)
[36D] 713:      DEC AL ;AL sooft erniedrigen bi
s Zeiger (DX) den
[HRG] 714:      JMP vergl123 ;Wert hat, um richti
ges Datenbit anzuzeigen
[W1E] 715: Zeigel23: STRING_ANZ ;zeige diesen Wert jet
zt an
[226] 716:      POP DX ;Cursor auf alte
[E27] 717:      CURS_SET ;Position setzen
[E96] 718: ;----- Datenbits Ende
[AEE] 719: ;letzte Abfrage, ob alles richtig (Stoppbit
)
[RKN] 720: The_Last3: MOV SI,Offset Eingabe3 ;es wurde
nur eine Zahl eingegeben (5-8)
[PD7] 721:      MOV AL,[SI] ;in AL übergeben
[1G5] 722:      SUB AL,30h ;minus 30hex
[Y74] 723:      CMP AL,1 ;wenn=1
```

```
[489] 724:      JAE Stobit_x ;springe Stoppbit
[X74] 725:      CMP AL,2 ;wenn<=2
[689] 726:      JBE Stobit_x ;springe Stoppbit
[4U9] 727:      JMP The_last3 ;nur bei Falsch-We
rten
[EFD] 728: Stobit_x: CURS_LES ;Cursorposition les
en
[LH6] 729:      PUSH DX ;sichere diese
[T4A] 730:      MOV DL,19 ;setze Cursor auf S
palte 19
[CZ5] 731:      MOV DH,4 ;und Zeile 4
[VP6] 732:      CURS_SET ;setze Cursor
[ERB] 733:      MOV DX,Offset S_Bit ;hole Text (
Stoppbit)
[BV7] 734:      CMP AL,1 ;war es eine Eins ?
[LXD] 735:      JE Zeigel234 ;dann zeige sofort
ein Stoppbit an
[RAB] 736:      INC DX ;erhöhe Zeiger (zwe
i Stoppbit)
[M3B] 737:      INC DX ;nochmals (wegen En
de-Zeichen $)
[WRD] 738: Zeigel234: STRING_ANZ ;zeige diesen Wert
jetzt an
[E17] 739:      POP DX ;Cursor auf alte
[AN7] 740:      CURS_SET ;Position setzen
[W86] 741: ;----- Stoppbits Ende
[L8L] 742: ;allerletzte Frage, ob insgesamt alles rich
tig eingegeben? (J/N)
[2KC] 743: Frage: MOV AH,0 ;hole Zeichen von T
astatur
[GY1] 744:      INT TAST
[HF6] 745:      CMP AL,'j' ;wenn klein ja
[505] 746:      JE Frage_Kon ;oder
[545] 747:      CMP AL,'J' ;groß JA
[Y4H] 748:      JE Frage_Kon ;dann gehe zur' Meld
ung Konfiguration sichern
[W45] 749:      CMP AL,'n' ;wie oben
[TA2] 750:      JE Fr_New
[AT1] 751:      CMP AL,'N'
[TA2] 752:      JE Fr_New
[35E] 753:      MOV DX,Offset Beep ;Fehlerton bei
Falscheingabe
[A8H] 754:      STRING_ANZ ;Fehlerton ausgeben
, da falsche Taste betätigt
[GVB] 755:      JMP Frage ;weitere Zeichen ei
nlesen
[RFE] 756: Fr_New: JMP Baud_Set ;N betätigt=neue Ei
ngabe
[FTM] 757: ;Frage, ob eingestellte Konfiguration gespe
ichert werden soll?
[4PR] 758: Frage_Kon: MOV DX,Offset Param_Set ;Text ho
len, soll Konfiguration gesichert
[6K9] 759:      STRING_ANZ ;werden und anzeige
n
[HEA] 760: Frage_Kol: MOV AH,0 ;Taste einlesen
[BS3] 761:      INT TAST
[E35] 762:      CMP AL,'j' ;klein ja
[6UC] 763:      JE Konfi_Sich ;dann Konfigurati
on sichern
[RE4] 764:      CMP AL,'J' ;groß JA
[6UC] 765:      JE Konfi_Sich ;dann Konfigurati
on sichern
[XV5] 766:      CMP AL,'n' ;klein nein
[CF6] 767:      JE Warte ;gehe zur DFÜ
[RE5] 768:      CMP AL,'N' ;groß NEIN
[CF6] 769:      JE Warte ;gehe zur DFÜ
[35E] 770:      MOV DX,Offset Beep ;Fehlerton be
i Falscheingabe
[WGH] 771:      STRING_ANZ ;Fehlerton ausgeben
, da falsche Taste betätigt
[TRE] 772:      JMP Frage_Kol ;keines von beiden
, auf J oder N warten
[V10] 773: ;
[Y5M] 774: Konfi_Sich: CALL Save_Konfig ;Sprung zum Si
chern der Konfiguration
[8M7] 775: ;----- Dateinamen eingeben
[7YB] 776: Warte: CALL Bild_Cls ;Bildschirm öschen
[774] 777:      MOV CL,0 ;Zähler
[F2E] 778:      MOV DI,Offset FCB ;Zeiger auf erst
es Zeichen im FCB
[V89] 779:      INC DI ;Zeiger um Eins erh
öhen
[VX6] 780: Wait1: MOV AL,32 ;32=Space
[VRK] 781:      MOV [DI],AL ;in FCB übertragen
um eventuell vorhandene Zeichen
zu überschreiben,
[26B] 782:      INC DI ;zu überschreiben,
DI erhöhen
[Z47] 783:      INC CL ;Zähler erhöhen
[HH8] 784:      CMP CL,8 ;8 Zeichen erreicht
?
[CS8] 785:      JE Dat_hol ;dann Textanzeige
[ZNA] 786:      JMP Wait1 ;nicht, dann weiter
machen
```



```

[V10] 787: ;
[X9L] 788: Dat_hol: MOV DX,Offset Dat_Name ;Text holen
      Dateiname eingeben
[AZ6] 789: STRING_ANZ ;und anzeigen
[6V7] 790: Entry: MOV CL,0 ;Zähler
[F2E] 791: MOV DI,Offset FCB ;Zeiger auf erst
      es Zeichen im FCB
[FXD] 792: Ein_Wei: INC DI ;Zeiger um eins erh
      öhen
[2KF] 793: MOV AH,0 ;auf Zeichen von Ta
      statur warten und in
[3G6] 794: INT TAST ;AL übergeben
[089] 795: CMP AL,0Dh ;Enter-Taste gedrüc
      kt ?
[1E7] 796: JE Dat_vor ;dann setze FCB
[9FA] 797: CMP AL,30h ;Zeichen kleiner 30
      hex ?
[VZA] 798: JB Warte ;nicht erlaubt, neu
      e Eingabe
[W2A] 799: CMP AL,7Ah ;Zeichen größer 7Ah
      ex ?
[TZA] 800: JA Warte ;nicht erlaubt, neu
      e Eingabe
[KG9] 801: Ueber: MOV AH,14 ;Zeichen aus AL
[3M5] 802: INT BIOS ;anzeigen
[HH8] 803: CMP CL,8 ;8 Zeichen erreicht
      ?
[1E7] 804: JE Dat_vor ;dann setze FCB
[2H6] 805: INC CL ;erhöhe Zähler
[BDA] 806: MOV [DI],AL ;übergebe Zeichen i
      n FCB
[DK9] 807: JMP Ein_Wei ;hole weitere Zeich
      en
[V10] 808: ;
[R9G] 809: Dat_vor: CMP CL,0 ;Enter-Taste, aber
      noch kein Zeichen
[10B] 810: JE Warte ;nicht erlaubt, neu
      e Eingabe
[MG8] 811: MOV DX,Offset FCB ;FCB existiert
[0ZE] 812: MOV AH,0Fh ;Öffne nun Datei au
      f aktuellem Laufwerk
[MY1] 813: INT DOS
[XOC] 814: CMP AL,255 ;Datei ist noch nic
      ht vorhanden
[16E] 815: JE DFU ;erzeuge diese auf
      aktuellem Laufwerk
[E6B] 816: MOV DX,Offset FCB ;Datei ist vorha
      nden
[LB8] 817: MOV AH,10h ;schließe Datei
[MY1] 818: INT DOS
[189] 819: CALL Bild_Cls ;lösche Bildschirm
[8CG] 820: MOV DX,Offset Schreib ;gebe Meldu
      ng Datei vorhanden aus
[T36] 821: STRING_ANZ ;anzeigen
[GBC] 822: Task1: MOV AH,0 ;hole Zeichen von T
      astatur
[GY1] 823: INT TAST
[X1F] 824: CMP AL,'j' ;klein ja, also Übe
      rschreiben erwünscht
[6C7] 825: JE DFU ;dann gehe zur DFU
[U3F] 826: CMP AL,'J' ;groß JA, also Über
      schreiben erwünscht
[BE7] 827: JE DFU ;dann gehe zur DFU
[5WD] 828: CMP AL,'n' ;klein nein, also n
      icht überschreiben
[KUD] 829: JE Waiting ;dann nochmals Date
      inamen angeben
[DYD] 830: CMP AL,'N' ;groß NEIN, also ni
      cht überschreiben
[KUD] 831: JE Waiting ;dann nochmals Date
      inamen angeben
[35E] 832: MOV DX,Offset Beep ;Fehlerton bei
      Falscheingabe
[A8H] 833: STRING_ANZ ;Fehlerton ausgeben
      , da falsche Taste betätigt
[66B] 834: JMP Task1 ;warte auf weitere
      Zeichen
[085] 835: Waiting: JMP Warte
[V10] 836: ;
[LFB] 837: DFU: CALL Bild_Cls ;Bildschirm löschen
[M58] 838: CURS_24 ;Cursor auf Zeile 2
      4
[BCE] 839: MOV DX,Offset Meldel ;gebe Meldung
      über Wartezeit
[4X3] 840: STRING_ANZ ;aus
[WW7] 841: CURS_0 ;Cursor auf Zeile 0
[V10] 842: ;
[81B] 843: ;----- ab hier Datenfernübertragung
[P6B] 844: MOV BX,OFFFh ;Zähler 1 für Prüfs
      chleife
[ZDA] 845: MOV CL,60 ;Zähler 2 für Prüfs
      chleife
[17F] 846: GER_Ber: MOV DX,RS+4 ;setze im Modemsteu
      erregister

```

```

[RGC] 847: MOV AL,00000001b ;Bit 0 auf 1 (Dat
      enendgerät bereit)
[TU1] 848: OUT DX,AL
[V10] 849: ;
[EYH] 850: Carr1: MOV DX,RS+6 ;eigenes Modem ist
      vorhanden, prüfe nun ob
[GUA] 851: IN AL,DX ;Gegenstelle sich m
      eldet
[61K] 852: TEST AL,00100000b ;wenn Bit 5 gese
      tzt, dann Modem betriebsbereit
[24C] 853: JNZ Aktiv ;(JNZ = Sprung wenn
      Bit gesetzt)
[S6F] 854: TEST AL,10000000b ;wenn Bit 7 gese
      tzt, dann Verbindung mit
[H97] 855: JNZ Aktiv ;Partnermodem
[OD7] 856: DEC BX ;Zähler minus Eins
[8P7] 857: CMP BX,0 ;Zähler auf Null
[FCE] 858: JE Carr_end ;dann erniedrige zw
      eiten Zähler
[1XD] 859: JMP Carr1 ;sonst warte weiter
      auf Gegenstelle
[V10] 860: ;
[RRD] 861: Carr_end: DEC CL ;zweiter Zähler min
      us Eins
[XUE] 862: CMP CL,0 ;wenn Null, dann si
      nd 60 Sekunden vergangen
[WZH] 863: JE Carr_end ;gebe Meldung inakt
      iv aus und beende Prozedur
[HCB] 864: JMP Carr1 ;nicht Null, warte
      weiter
[9NK] 865: Carr1_end: CALL Inaktiv ;Gegenstelle hat si
      ch nicht gemeldet
[E0H] 866: JMP Ende ;letzte Meldung aus
      geben und Prozedur beenden
[V10] 867: ;
[2FL] 868: Aktiv: MOV BX,0 ;BX (Attribut) für
      folgende Bildoperation auf Null
[RW9] 869: CALL Bild_Cls ;Bildschirm löschen
[7ZH] 870: CALL Melde_DFU ;Meldung Datenüber
      ragung aktiv in Zeile 24
[WW7] 871: CURS_0 ;Cursor auf Zeile 0
[V10] 872: ;
[CPK] 873: Lese_ein: MOV DX,Offset DTA ;DTA setzen für
      Zeichenoperation
[7X1] 874: MOV AH,1Ah
[MY1] 875: INT DOS
[BWK] 876: MOV DI,Offset DTA ;Zeiger auf ers
      tes Zeichen für Diskoperation
[K87] 877: MOV CX,0 ;Zähler auf Null
[V10] 878: ;
[YKG] 879: Lese_ein1: MOV DX,RS+6 ;prüfe, ob im Modem
      statusregister
[7BE] 880: IN AL,DX ;Bit 7 immer noch g
      esetzt ist (DFU aktiv)
[N7G] 881: TEST AL,100000000b ;JZ = Sprung w
      enn Null, also Bit nicht mehr
[VTL] 882: JZ Sicherend ;gesetzt, beende Pr
      ozedur, aber noch Datei sichern
[V10] 883: ;
[UPC] 884: MOV DX,RS+5 ;Partnermodem hat s
      ich gemeldet
[B2G] 885: IN AL,DX ;prüfe, ob Daten im
      Empfangspuffer vorhanden
[8L7] 886: TEST AL,00000001b ;sind (Bit 0=1)
[YE6] 887: JZ Gebe_Aus ;keine Daten
[XX4] 888: MOV DX,RS ;Daten OK
[CN6] 889: IN AL,DX ;lies Daten ein
[L2H] 890: MOV [DI],AL ;sichere diese für
      spätere Diskoperationen
[EN5] 891: INC DI ;Sicherung+1
[905] 892: INC CX ;Zähler+1
[00C] 893: CMP CX,10240 ;Zeichen für 80 Sät
      ze erreicht ?
[WFB] 894: JE Vorh_DatS ;dann schon abspeic
      hern
[1AC] 895: ;----- Bildschirmsteuerung-Datenempfang
[H7L] 896: ;wenn besondere Zeichen empfangen werden, m
      üssen DI und CX
[EVA] 897: ;erhöht oder vermindert werden
[V10] 898: ;
[WDM] 899: Ret_ein: CMP AL,0Dh ;wenn Enter empfang
      en, dann besondere Behandlung
[7G2] 900: JE Scroll
[BHK] 901: Crsr_leftin: CMP AL,08h ;Left wie Rückwärts
      schritt behandeln
[6BD] 902: JE Backspace ;(Rückwärtsschrit
      t genauso)
[V10] 903: ;
[OTD] 904: Zei_an: MOV AH,14 ;sonst Zeichen anz
      eigen
[AW1] 905: INT BIOS
[V10] 906: ;
[75F] 907: ;----- Prüfe jetzt ,ob Zeile 24 erreicht
      und Spalte 1

```



```

[DX1] 908:          PUSH CX
[1M9] 909:          CURS_LES          ;Cursorposition les
en
[XD1] 910:          POP CX
[Z3H] 911:          CMP DH,24          ;Zeile 24 erreicht?
(wenn Zeichen eingegeben in
[YNM] 912:          JE Bildscroll ;DFÜ aber keine Ent
er-Taste, muß reagiert werden)
[ANC] 913:          JMP Lese_einl ;weitere Zeichen ei
nlesen
[VW0] 914:          ;
[H5B] 915:          ;----- Backspace (Rückwärtsschritt)
[HWA] 916:          Backspace: DEC DI          ;Zeiger zweifach
[NL6] 917:          DEC DI          ;zurückzählen
[524] 918:          DEC CX          ;Zähler
[KB3] 919:          DEC CX          ;auch
[GM2] 920:          JMP Zei_an
[NP4] 921:          ;----- Enter-Taste
[H5C] 922:          Scroll: MOV AH,14          ;zeige Daten sofort
an
[723] 923:          INT BIOS          ;an
[5E7] 924:          PUSH CX          ;Zähler sichern
[1M9] 925:          CURS_LES          ;Cursorposition les
en
[BL6] 926:          POP CX          ;Zähler zurück
[V3A] 927:          CMP DH,23          ;Zeile 23 erreicht+
Enter?
[6WE] 928:          JAE Bildscroll ;ja, dann scrolle B
ild nach oben
[R6F] 929:          JMP Lese_einl ;nicht erreicht, da
nn weiter einlesen
[V10] 930:          ;
[5LR] 931:          ;Bild wird gescrollt bei Erreichen von Zeil
e 23 und Spalte 80 oder Zeile 23+Enter
[P6K] 932:          Bildscroll: PUSH DX          ;sichere Zeilen- un
d Spaltenposition
[HA9] 933:          CALL Bild_Cls24 ;Zeile 24 lösch
en
[6MK] 934:          MOV AL,0Ah          ;Zeilenvorschub (sp
ringe damit zur nächsten Zeile)
[VLD] 935:          MOV AH,14          ;schreibe Zeichen a
uf Bildschirm und
[3V9] 936:          INT BIOS          ;bewege Cursor vorw
ärts
[DDD] 937:          CALL Melde_DFÜ ;bringe erneut M
eldung in Zeile 24
[WSC] 938:          POP DX          ;hole Zeilen und Sp
altenposition
[6SF] 939:          DEC DH          ;minus eine Zeile (
darf nicht über 24 hinaus)
[RZ6] 940:          CURS_SET          ;setze Cursor
[BWC] 941:          ;direkte Sprünge, weil JE und JZ zu kurz
[F27] 942:          Retour: JMP Lese_einl
[K07] 943:          Vorh_Dats: JMP Dat_Save1
[N87] 944:          Sicherend: JMP Dat_Save
[V10] 945:          ;
[YVL] 946:          Gebe_aus: CALL Pruefe          ;prüfe, ob Baustein
ein Zeichen senden kann
[SRG] 947:          MOV AH,0Bh          ;Baustein bereit, p
rüfe ob Zeichen überhaupt
[AL8] 948:          INT DOS          ;eingegeben wurde
[LF5] 949:          CMP AL,0          ;wenn nicht,
[TAL] 950:          JZ Retour          ;dann prüfe, ob wie
der ein Zeichen empfangen wird
[HS9] 951:          MOV AH,0          ;Zeichen in Puffer
nach AL
[6Y5] 952:          INT TAST          ;übergeben
[FPV] 953:          ;wenn besondere Tasten gedrückt werden, muß
darauf reagiert werden
[2U8] 954:          CMP AH,68          ;wurde [F10] gedrück
t ?
[D3H] 955:          JE Sicherend ;ja, dann Ende, vor
her aber Datei abspeichern
[SH8] 956:          CMP AH,67          ;wurde [F9] gedrück
t ?
[CFD] 957:          JE Cancel          ;sende CTRL-X (ungü
ltig) = DFÜ beenden
[VX0] 958:          ;
[PUD] 959:          Ret_Tast: CMP AL,0Dh          ;Zeichencode für [E
nter]
[5CH] 960:          JE Line_Feed ;ja, dann bearbeite
n (noch Linefeed senden)
[MUF] 961:          Crsr_left: CMP AH,75          ;Scancode für Taste
Cursor-Left
[X3A] 962:          JE Crsr_Steul ;ja, dann bearbeit
en
[PL3] 963:          ;----- Sende
[SB9] 964:          Sende_Line: MOV DX,RS          ;Zeichen in
[F59] 965:          OUT DX,AL          ;Ausgabekanal=sende
n
[C2B] 966:          JMP Retour          ;weitere Zeichen ei
nlesen
[1HG] 967:          ;sende DFÜ-Beenden-Zeichen, lese aber noch
weiter ein

```

```

[CDA] 968:          Cancel: MOV AL,18h          ;Wert für Cancel
[GT9] 969:          JMP Sende_Line ;sende jetzt Cancel
[V10] 970:          ;
[COE] 971:          Line_Feed: MOV DX,RS          ;Enter-Taste wurde
gedrückt
[UUE] 972:          MOV AL,0Dh          ;es muß Wagenrückkla
uf gesendet werden
[6LB] 973:          OUT DX,AL          ;sende jetzt Wagenr
ücklauf
[L6C] 974:          ;nun muß noch Linefeed gesendet werden
[1XG] 975:          CALL Pruefe          ;pruefe, ob Baustei
n ein Zeichen senden kann
[078] 976:          MOV AL,0Ah          ;Wert für Linefeed
[ZMA] 977:          JMP Sende_line ;sende jetzt Linefe
ed
[V10] 978:          ;
[POC] 979:          Crsr_Steul: MOV AL,08h          ;Wert für Backspace
[WZA] 980:          JMP Sende_line ;sende jetzt Bac
kspace
[U17] 981:          ;----- Sicherung der DFÜ
[5Y8] 982:          ;----- Sende CTRL-X = DFÜ beenden
[4R5] 983:          DAT_Save: CALL Pruefe
[MRA] 984:          MOV AL,18h          ;Wert für Cancel=En
de DFÜ
[GA5] 985:          MOV DX,RS          ;Zeichen in
[7S8] 986:          OUT DX,AL          ;Ausgabekanal=sende
n
[OVL] 987:          ;schreibe noch Dateiendezeichen (CTRL-Z) in
zu sichernde Datei
[GA8] 988:          MOV AL,1Ah          ;Dateiendezeichen
[XA2] 989:          MOV [DI],AL
[V10] 990:          ;
[5GL] 991:          Dat_Save1: MOV AH,0          ;sichere Dateiendez
eichen (wenn vorhanden)
[1X1] 992:          PUSH AX
[FSG] 993:          CALL Pruefe          ;prüfe, ob Baustein
ein Zeichen senden kann
[EB5] 994:          MOV AL,13h          ;CTRL-S (X-OFF)
[C6E] 995:          MOV DX,RS          ;sende jetzt Ausgab
e-Anhalten-Zeichen
[TU1] 996:          OUT DX,AL
[C2B] 997:          Open: MOV DX,Offset FCB ;Datei öffnen
[CX1] 998:          MOV AH,0Fh
[MY1] 999:          INT DOS
[6UA] 1000:          CMP AL,255          ;Datei noch nicht v
orhanden
[TKB] 1001:          JE Create          ;springe zur erzeug
en Datei
[M24] 1002:          MOV DX,Offset FCB
[BK8] 1003:          MOV CX,80          ;Anzahl der Sätze (
80)
[6AB] 1004:          MOV AH,28h          ;wahlfrei 80 Sätze
schreiben
[MY1] 1005:          INT DOS
[V10] 1006:          ;
[M24] 1007:          MOV DX,Offset FCB
[KL8] 1008:          MOV AH,10h          ;Datei schließen
[MY1] 1009:          INT DOS
[V10] 1010:          ;
[UOH] 1011:          POP AX          ;wenn [F10], dann m
uß hier ein Dateiendezeichen
[LDB] 1012:          CMP AL,1Ah          ;stehen, prüfe, ob
das der Fall
[VZ8] 1013:          JE Ende          ;ja, dann beende DF
Ü
[M8K] 1014:          CALL Pruefe          ;nein, prüfe, ob Ba
ustein ein Zeichen senden kann
[B95] 1015:          MOV AL,11h          ;CTRL-Q (X-ON)
[EYE] 1016:          MOV DX,RS          ;sende jetzt Ausgab
e-Fortsetzen-Zeichen
[TU1] 1017:          OUT DX,AL
[FB3] 1018:          JMP Lese_ein
[V10] 1019:          ;
[5ZB] 1020:          Create: MOV DX, Offset FCB ;erzeuge Datei
[B32] 1021:          MOV AH, 16h
[MY1] 1022:          INT DOS
[2A2] 1023:          JMP Open
[VX0] 1024:          ;
[ELE] 1025:          ENDE: CALL Inaktiv          ;Meldung DFÜ inakti
v ausgeben
[9DH] 1026:          CALL Bild_Cls24 ;Bildschirmattribu
te in Zeile 24 löschen
[RW9] 1027:          CALL Bild_Cls          ;Bildschirm löschen
[SE0] 1028:          RET
[V10] 1029:          ;
[CY3] 1030:          Programm ENDP
[V10] 1031:          ;
[W07] 1032:          ;----- Unterprogramme
[814] 1033:          ;----- Inaktiv
[914] 1034:          Inaktiv PROC
[PB7] 1035:          MOV DH,24          ;Cursor in Zeile 24
[FU5] 1036:          MOV DL,0          ;und Spalte 0
[H15] 1037:          CURS_SET          ;setzen

```



```

[YYE] 1038:      MOV DX,Offset Melde3 ;Zeiger auf M
eldung DFÜ inaktiv
[6AB] 1039:      STRING_ANZ ;Meldung in Zeile 24 an
zeigen
[75C] 1040:      ;sende jetzt noch Ende der Übertragung
[72M] 1041:      Inaktivp: CALL Pruefe ;prüfe, ob Baustein
ein Zeichen senden kann
[SU1] 1042:      MOV DX,RS
[MPE] 1043:      MOV AL,18h ;Wert für Cancel (E
nde der Übertragung)
[RB8] 1044:      OUT DX,AL ;(Cancel=ungültig)
[FFA] 1045:      ;Schleife zum Lesen der Meldung
[R05] 1046:      MOV CL,10 ;Zähler 1
[7X7] 1047:      Vel: MOV BX,0FFFFh ;Zähler 2
[PZ6] 1048:      Ve2: DEC BX ;minus Eins
[985] 1049:      CMP BX,0 ;wenn Null,
[3R9] 1050:      JE Ve3 ;dann Zähler 1 minu
s Eins
[DED] 1051:      JMP Ve2 ;nicht Null, weiter
runterzählen
[A39] 1052:      Ve3: DEC CL ;BX=0, CL minus Ein
s
[SRE] 1053:      CMP CL,0 ;solange CL nicht N
ull ist, weiter BX DEC.
[41P] 1054:      JE Inaktiv_off ;wenn CL gleich Nul
l, dann Rücksprung aus Unterprozedur
[XT6] 1055:      JMP Vel ;weitermachen
[CK4] 1056:      Inaktiv_off: RET
[SE4] 1057:      Inaktiv ENDP
[D04] 1058:      ;----- Fehler
[KZ2] 1059:      False PROC
[0HA] 1060:      CURS_LES ;Cursorposition lesen
[68A] 1061:      PUSH DX ;sichere Cursorposition
[9C8] 1062:      PUSH BX ;Zähler sichern (BL)
[ZA8] 1063:      ;----- Fehlermeldung und Ton
[LZB] 1064:      MOV DX,Offset Beep ;Aufmerksamkei
tston
[2YC] 1065:      STRING_ANZ ;ausgeben ( gibt au
ch Töne aus)
[F3D] 1066:      MOV DL,0 ;Fehlermeldung ausg
eben in Spalte 0
[OK5] 1067:      MOV DH,24 ;und Zeile 24
[YM9] 1068:      CURS_SET ;Cursor positionier
en
[SDE] 1069:      MOV DI,Offset Falsch ;Zeiger auf e
rstes Zeichen
[FDG] 1070:      Ausgeb: MOV AL,[DI] ;erstes Zeichen in
AL übergeben
[67C] 1071:      CMP AL,24h ;letztes Zeichen ($
) erreicht?
[DF2] 1072:      JE Weiter
[327] 1073:      INC DI ;Zeiger erhöhen
[GUB] 1074:      MOV CX,1 ;jeweils ein Zeiche
n anzeigen
[BC5] 1075:      MOV BL,112 ;Revers ein
[PM1] 1076:      MOV AH,9
[AW1] 1077:      INT BIOS
[DC8] 1078:      INC DL ;Spalten erhöhen
[LAB] 1079:      CURS_SET ;und Cursor positio
nieren
[5N2] 1080:      JMP Ausgeb
[KU8] 1081:      ;----- Verzögerungsschleife
[T88] 1082:      Weiter: MOV AL,5 ;Zähler 1
[KR7] 1083:      V1: MOV BX,0FFFFh ;Zähler 2
[L4H] 1084:      V2: DEC BX ;siehe Weiter Zähls
chleife in Unterprozedur
[3L5] 1085:      CMP BX,0 ;inaktiv
[M41] 1086:      JE V3
[E91] 1087:      JMP V2
[7R2] 1088:      V3: DEC AL
[EM1] 1089:      CMP AL,0
[OL3] 1090:      JE Weiter_on
[891] 1091:      JMP V1
[LT4] 1092:      ;----- Revers aus
[82D] 1093:      Weiter_on: MOV DL,0 ;Fehlermeldung wurd
e revers
[PX5] 1094:      MOV DH,24 ;ausgegeben
[CGA] 1095:      CURS_SET ;überschreibe diese
nun
[H1B] 1096:      MOV AL,32 ;mit normalen Zeich
en (Space)
[Z2G] 1097:      MOV CX,46 ;lade CX mit Anzahl
zu schreibender Zeichen
[HB8] 1098:      MOV BL,2h ;lade BL mit Attrib
ut
[5D5] 1099:      MOV AH,9 ;Funktion 9
[AW1] 1100:      INT BIOS
[V10] 1101:      ;
[PT8] 1102:      POP BX ;Zähler zurückholen
[Y09] 1103:      POP DX ;Cursor auf alten Wert
[8M5] 1104:      DEC DL ;minus Eins
[KBE] 1105:      CMP BL,2 ;sind zwei falsche Zeich
en eingegeben
[98G] 1106:      JE Decre ;worden, wenn ja dann Cu
rsorbehandlung

```

```

[CK9] 1107:      False_1: CURS_SET ;Cursor setzen
[RMC] 1108:      MOV AL,32 ;überschreibe falsches Z
eichen
[6P1] 1109:      MOV AH,10
[7AE] 1110:      MOV CX,1 ;Anzahl (1) zu überschre
ibender Zeichen
[AW1] 1111:      INT BIOS
[KBE] 1112:      CMP BL,2 ;sind zwei falsche Zeich
en eingegeben
[98G] 1113:      JE Decre ;worden, wenn ja dann Cu
rsorbehandlung
[845] 1114:      RET ;neuer Lauf
[WAC] 1115:      Decre: MOV DL,21 ;setze Cursor in Spalte
21
[LH5] 1116:      MOV DH,22 ;und Zeile 22
[68A] 1117:      PUSH DX ;sichere Cursorposition
[3K9] 1118:      CURS_SET ;setze Cursor jetzt
[USF] 1119:      MOV AL,32 ;überschreibe falsches Z
eichen (Space)
[ZD5] 1120:      MOV AH,10 ;Funktion 10
[BAE] 1121:      MOV CX,2 ;Anzahl (2) zu überschre
ibender Zeichen
[AW1] 1122:      INT BIOS
[ZEL] 1123:      POP DX
[7W7] 1124:      CURS_SET ;setze Cursor
[SE0] 1125:      RET
[5H2] 1126:      FALSE ENDP
[FTF] 1127:      ;prüfe, ob Baustein bereit, ein Zeichen zu
senden
[4D3] 1128:      Pruefe PROC
[UXF] 1129:      Probef: MOV DX,RS+5 ;prüfe, ob Baustein
bereit ist, ein
[P6A] 1130:      IN AL,DX ;neues Zeichen zu s
enden
[3A5] 1131:      TEST AL,00100000b ;Bit 5=1
[GMD] 1132:      JZ Probef ;noch nicht bere
it, warte weiter
[SE0] 1133:      RET
[RC3] 1134:      Pruefe ENDP
[PM8] 1135:      ;----- Konfiguration sichern
[3U4] 1136:      Save_Konfig PROC
[524] 1137:      ;----- DTA setzen
[1CE] 1138:      MOV DX,Offset DTA_1 ;DTA setzen fü
r Konfiguration
[7X1] 1139:      MOV AH,1ah
[MY1] 1140:      INT DOS
[V10] 1141:      ;
[06D] 1142:      MOV DI,Offset DTA_1 ;Zeiger auf er
stes Zeichen
[LU9] 1143:      MOV DX,RS+3 ;hole 3. Wert
zuerst !!!
[9CD] 1144:      IN AL,DX ;vom Leitungss
steuerregister
[B14] 1145:      MOV AH,0 ;AH=0
[GM7] 1146:      PUSH AX ;sichere 3. We
rt
[F1N] 1147:      ;setze Bit 7 um Baudratengenerator zu progr
ammieren (auszulesen)
[2EC] 1148:      MOV DX,RS+3 ;Leitungssteuerregi
ster Bit 7=1
[EX1] 1149:      MOV AL,80h
[7Y4] 1150:      OUT DX,AL ;setzen
[V10] 1151:      ;
[977] 1152:      MOV DX,RS ;hole ersten Wert
[CKA] 1153:      IN AL,DX ;vom Baudraten-Regi
ster (Low)
[V10] 1154:      ;
[8NE] 1155:      MOV [DI],AL ;in Konfiguration z
wischenspeichern
[327] 1156:      INC DI ;Zeiger erhöhen
[V10] 1157:      ;
[NZ7] 1158:      MOV DX,RS+1 ;hole zweiten Wert
[NYA] 1159:      IN AL,DX ;vom Baudraten-Regi
ster (High)
[V10] 1160:      ;
[8NE] 1161:      MOV [DI],AL ;in Konfiguration z
wischenspeichern
[327] 1162:      INC DI ;Zeiger erhöhen
[PRD] 1163:      ;lösche Bit 7 im Leitungssteuerregister
[WDC] 1164:      MOV DX,RS+3 ;Leitungssteuerregi
ster Bit 7=0
[EX1] 1165:      MOV AL,00h
[7Y4] 1166:      OUT DX,AL ;setzen
[V10] 1167:      ;
[2FA] 1168:      POP AX ;hole dritten Wert
zurück
[8NE] 1169:      MOV [DI],AL ;in Konfiguration z
wischenspeichern
[327] 1170:      INC DI ;Zeiger erhöhen
[4SG] 1171:      MOV DX,RS+3 ;Leitungssteuerregi
ster vorbelegten Wert
[UP5] 1172:      OUT DX,AL ;neu setzen
[9XM] 1173:      ;Hier werden vom Bildschirm die bisher eing
egebenen Daten gelesen

```



```

[SYB] 1174:      CURS_0      ;setze Cursor auf Zeile/Spalte 0
[L8G] 1175: Insert: MOV AH,8      ;Zeichen an Cursorposition lesen
[AW1] 1176:      INT BIOS
[EYE] 1177:      MOV [DI],AL      ;gelesenes Zeichen zwischenspeichern
[K6C] 1178:      CMP AL,'N'      ;letztes Zeichen (N) erreicht?
[WB9] 1179:      JE Disk_Save    ;dann abspeichern
[058] 1180:      CMP DL,80      ;Spalte 80 erreicht?
[A99] 1181:      JE And_Zeile    ;dann Zeilen erhöhen
[327] 1182:      INC DI          ;Zeiger erhöhen
[DC8] 1183:      INC DL          ;Spalten erhöhen
[9LB] 1184:      CURS_SET      ;Cursorposition neu setzen
[HU8] 1185:      JMP Insert      ;nein, weiterlesen
[V10] 1186:      ;
[MAE] 1187: And_Zeile: MOV DL,0    ;Spalten auf Wert N null setzen
[9M6] 1188:      INC DH          ;Zeilen erhöhen
[SZ6] 1189:      CURS_SET      ;Cursor setzen
[MP2] 1190:      JMP Insert
[H14] 1191:      ;----- Disk I/O
[V7G] 1192: Disk_Save: INC DI      ;Zeiger erhöhen, schreiben in Datei
[G6D] 1193:      MOV AL,1Ah      ;Control-Z (1Ah) als Dateiendezeichen
[K77] 1194:      MOV [DI],AL      ;übergebe CTRL-Z
[AM8] 1195:      ;----- Wartemeldung ausgeben
[RW9] 1196:      CALL Bild_Cls    ;Bildschirm löschen
[WV7] 1197:      CURS_0      ;Cursor auf Zeile 0
[LVE] 1198:      MOV DX,Offset Warten ;Text Moment bitte... anzeigen
[0F2] 1199:      STRING ANZ
[2M5] 1200:      ;----- Datei anlegen
[ASA] 1201:      MOV DX,Offset FCB_Save ;Datei anlegen
[E9B] 1202:      MOV AH,16h      ;auf dem aktuellen Laufwerk
[MY1] 1203:      INT DOS
[V10] 1204:      ;
[V16] 1205:      MOV DX,Offset FCB_Save
[6A8] 1206:      MOV CX,4      ;Anzahl der Sätze (4)
[XZB] 1207:      MOV AH,28h      ;wahlfrei vier Sätze schreiben
[MY1] 1208:      INT DOS
[V10] 1209:      ;
[V16] 1210:      MOV DX,Offset FCB_Save
[F1H] 1211:      MOV AH,10h      ;Datei schließen auf dem aktuellen Laufwerk
[MY1] 1212:      INT DOS
[V10] 1213:      ;
[SE0] 1214:      RET
[7N4] 1215: Save_Konfig ENDP
[F18] 1216:      ;----- Konfiguration laden
[HL4] 1217: Load_Konfig PROC
[524] 1218:      ;----- DTA setzen
[1CE] 1219:      MOV DX,Offset DTA_1 ;DTA setzen für Konfiguration
[7X1] 1220:      MOV AH,1Ah
[MY1] 1221:      INT DOS
[MF4] 1222:      ;----- Datei laden
[7Y9] 1223:      MOV DX,Offset FCB_Load ;Datei öffnen
[Y9B] 1224:      MOV AH,0Fh      ;auf dem aktuellen Laufwerk
[MY1] 1225:      INT DOS
[MXD] 1226:      CMP AL,0      ;ist die Datei überhaupt vorhanden?
[LX9] 1227:      JNZ End_LKo      ;nein, dann Rücksprung
[V10] 1228:      ;
[FZ5] 1229:      MOV DX,Offset FCB_Load
[6A8] 1230:      MOV CX,4      ;Anzahl der Sätze (4)
[9MA] 1231:      MOV AH,27h      ;wahlfrei vier Sätze lesen
[MY1] 1232:      INT DOS
[V10] 1233:      ;
[FZ5] 1234:      MOV DX,Offset FCB_Load
[47H] 1235:      MOV AH,10h      ;Datei schließen auf dem aktuellen Laufwerk
[MY1] 1236:      INT DOS
[V10] 1237:      ;
[06D] 1238:      MOV DI,Offset DTA_1 ;Zeiger auf erstes Zeichen
[000] 1239:      ;
[18M] 1240:      ;setze Bit 7, um Baudratengenerator zu programmieren (auszulesen)
[2EC] 1241:      MOV DX,RS+3      ;Leitungssteuerregister Bit 7=1

```

```

[EX1] 1242:      MOV AL,80h
[7Y4] 1243:      OUT DX,AL      ;setzen
[V10] 1244:      ;
[V42] 1245:      MOV AL,[DI]
[327] 1246:      INC DI          ;Zeiger erhöhen
[SU1] 1247:      MOV DX,RS
[7UA] 1248:      OUT DX,AL      ;vom Baudraten-Register (Low)
[V10] 1249:      ;
[V42] 1250:      MOV AL,[DI]
[327] 1251:      INC DI          ;Zeiger erhöhen
[122] 1252:      MOV DX,RS+1
[G6B] 1253:      OUT DX,AL      ;vom Baudraten-Register (High)
[V10] 1254:      ;
[WDC] 1255:      MOV DX,RS+3      ;Leitungssteuerregister Bit 7=0
[EX1] 1256:      MOV AL,00h
[7Y4] 1257:      OUT DX,AL      ;setzen
[V10] 1258:      ;
[V42] 1259:      MOV AL,[DI]
[327] 1260:      INC DI          ;Zeiger erhöhen
[ZPG] 1261:      MOV DX,RS+3      ;Leitungssteuerregister, vorbelegten Wert
[7Y4] 1262:      OUT DX,AL      ;setzen
[92M] 1263:      ;hier werden vom Bildschirm die bisher eingegebenen Daten gelesen
[G5C] 1264:      MOV DX,Offset Konfig_Mask ;Text anzeigen
[0F2] 1265:      STRING ANZ
[B04] 1266:      MOV CL,80 ;Zähler
[NT9] 1267:      Verm: INC DI      ;DTA-Zeiger erhöhen
[K28] 1268:      DEC CL      ;Zähler minus Eins
[025] 1269:      CMP CL,0 ;Zähler=0?
[ZFG] 1270:      JE LES_Konfi      ;ja, dann zeige geladene Konfiguration an
[E5A] 1271:      JMP Verm          ;nein: Zähler minus Eins
[H7E] 1272:      Les_Konfi: MOV AL,[DI] ;aus DTA, DI in AL übergeben
[D7E] 1273:      MOV AH,14      ;anzeigen und Cursor um Eins vorbewegen
[AW1] 1274:      INT BIOS
[AXA] 1275:      CMP AL,'N'      ;letztes Zeichen (N) erreicht?
[GE9] 1276:      JE End_LKo      ;ja, dann Rücksprung
[EW8] 1277:      INC DI          ;nein, weiter einlesen
[BV3] 1278:      JMP Les_Konfi
[9V3] 1279:      End_LKo: RET
[TL4] 1280: Load_Konfig ENDP
[DH7] 1281:      ;----- Bildschirm löschen
[AM3] 1282: Bild_Cls PROC
[DXA] 1283:      CURS_0      ;Cursor auf Spalte 0 und Zeile 0
[505] 1284:      MOV AL,32      ;32=Space
[6P1] 1285:      MOV AH,10
[K38] 1286:      MOV CX,2000      ;2000 Zeichen sind zu
[BR5] 1287:      INT BIOS      ;löschen
[DXA] 1288:      CURS_0      ;Cursor auf Spalte 0 und Zeile 0
[SE0] 1289:      RET
[XL3] 1290: Bild_Cls ENDP
[V10] 1291:      ;
[CE4] 1292: Bild_Cls24 PROC
[4X6] 1293:      PUSH CX      ;sichere Zähler
[DNB] 1294:      CURS_24      ;Cursor auf Zeile 24 und Spalte 0
[BV4] 1295:      MOV AL,32      ;32=Space
[PM1] 1296:      MOV AH,9
[GBF] 1297:      MOV BL,00001111b ;Attribut in BL für Zeichen normal hell
[5K8] 1298:      MOV CX,80      ;80 Zeichen löschen
[ZN9] 1299:      INT BIOS      ;Cursor bleibt in Zeile 24
[GU8] 1300:      POP CX      ;Zähler wieder zurück
[SE0] 1301:      RET
[5D4] 1302: Bild_Cls24 ENDP
[V10] 1303:      ;
[423] 1304: Melde_DFU PROC
[RLC] 1305:      MOV DL,0      ;Meldung ausgeben in Spalte Null
[RL5] 1306:      MOV DH,24      ;und Zeile 24
[XF9] 1307:      CURS_SET      ;Cursor positionieren
[4AE] 1308:      MOV SI,Offset Melde2 ;Zeiger auf erstes Zeichen
[LVG] 1309: Ausgeben: MOV AL,[SI] ;erstes Zeichen in AL übergeben
[G4B] 1310:      CMP AL,24h      ;letztes Zeichen ($) erreicht?

```



```

[CN3] 1311:      JE Ende_ausg
[ZP6] 1312:      INC SI      ;Zeiger erhöhen
[DX1] 1313:      PUSH CX
[75B] 1314:      MOV CX,1    ;jeweils ein Zeiche
      n anzeigen
[XC5] 1315:      MOV BL,112  ;Revers ein
[PM1] 1316:      MOV AH,9
[AW1] 1317:      INT BIOS
[5A7] 1318:      INC DL      ;Spalten erhöhen
[XD1] 1319:      POP CX
[XF9] 1320:      CURS_SET    ;Cursor positionier
      en
[CC3] 1321:      JMP Ausgeben
[984] 1322: Ende_ausg: RET
[HA4] 1323: Melde_DFU ENDP
[V10] 1324: ;
[SA3] 1325: Code          ENDS
[3M3] 1326:          END Programm
[000] 1327:

```

Listing 1. Der Quellcode von »STrans« beweist, daß das Programm ausschließlich in Assemblersprache geschrieben ist

```

[6F7] 0: 4D 5A 36 00 26 00 01 00 20 00 00 00 FF FF 00 00
[YC7] 1: 40 00 C1 A1 00 00 FC 03 1E 00 00 00 01 00 07 00
[B27] 2: FC 03 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 3: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      : : : : : : : : : : : : : : : :
[007] 35: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[36] 36: 20 20 20 20 20 20 35 30 24 20 20 20 20 20 20
[E57] 37: 20 20 20 20 20 20 20 20 20 20 20 20 37 35 24 20
[857] 38: 20 20 20 20 20 20 31 31 30 24 20 20 20 20 31 33
[397] 39: 34 2C 35 24 20 20 20 20 20 20 20 20 31 35 30 24
[U47] 40: 20 20 20 20 20 20 33 30 24 20 20 20 20 20 20
[877] 41: 20 36 30 30 24 20 20 20 20 20 20 31 32 30 30 24
[E67] 42: 20 20 20 20 20 20 31 38 30 24 20 20 20 20 32
[847] 43: 30 30 30 24 20 20 20 20 20 20 20 20 20 20 20
[H57] 44: 20 20 20 20 32 34 30 30 24 20 20 20 20 20 33
[A67] 45: 36 30 30 24 20 20 20 20 20 20 34 38 30 30 24 20
[X47] 46: 20 20 20 20 20 20 37 32 30 30 24 20 20 20 20
[HF7] 47: 39 36 30 30 24 20 20 20 6B 65 69 6E 65 24 20 75
[CF7] 48: 6E 67 65 72 61 64 65 24 20 20 20 67 65 72 61 64
[F57] 49: 65 24 20 20 20 20 20 20 35 24 20 20 20 20 20
[C47] 50: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 24
[G37] 51: 20 20 20 20 20 20 20 20 20 20 24 20 20 20 20
[M37] 52: 20 20 20 20 20 24 20 20 20 20 20 20 20 20 20
[457] 53: 24 20 20 20 20 20 20 20 20 20 36 24 20 20 20
[B57] 54: 20 20 20 20 20 20 37 24 20 20 20 20 20 20 20
[WB8] 55: 20 38 24 31 24 32 24 C9 CD CD CD CD CD CD CD
[BV7] 56: CD 20 56 6F 6E 20 49 68 6E 65 6E 20 65 69 6E 67
[3F8] 57: 65 73 74 65 6C 6C 74 3A 20 CD CD CD CD CD CD
[N27] 58: CD CD CD CD CD BB 0A 0D BA 20 20 20 20 20 20 20
[X47] 59: 20 20 20 20 20 20 20 20 20 20 20 20 20 42 61
[357] 60: 75 64 20 20 20 20 20 20 20 20 20 20 20 20 20
[AG7] 61: 20 20 20 20 20 20 BA 0A 0D BA 20 20 20 20 20 20
[247] 62: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 50
[087] 63: 61 72 69 74 84 74 20 20 20 20 20 20 20 20 20
[LH7] 64: 20 20 20 20 20 20 20 20 BA 0A 0D BA 20 20 20 20
[G37] 65: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[7E7] 66: 44 61 74 65 6E 62 69 74 20 20 20 20 20 20 20
[M97] 67: 20 20 20 20 20 20 20 20 BA 0A 0D BA 20 20 20 20
[G37] 68: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[9C7] 69: 20 53 74 6F 70 70 62 69 74 20 20 20 20 20 20
[X77] 70: 20 20 20 20 20 20 20 20 BA 0A 0D C8 CD CD CD
[849] 71: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 72: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[3R8] 73: CD CD CD CD CD CD CD CD CD CD BC 0A 0D 53 74 69
[CR7] 74: 6D 6D 65 6E 20 64 69 65 73 65 20 41 6E 67 61 62
[2C8] 75: 65 6E 20 3F 20 4A 2F 4E 20 24 C9 CD CD CD CD
[849] 76: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 77: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 78: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[6A8] 79: CD CD CD CD CD CD BB 0A 0D BA 20 44 69 65 20 66
[AP7] 80: 6F 6C 67 65 6E 64 65 20 42 61 75 64 72 61 74 65
[6P7] 81: 6E 20 73 74 65 68 65 6E 20 7A 75 72 20 44 61 74
[YP7] 82: 65 6E 81 62 65 72 74 72 61 67 75 6E 67 20 62 65
[TP7] 83: 72 65 69 74 20 BA 0A 0D BA 20 53 69 65 20 6B 94
[AS7] 84: 6E 6E 65 6E 20 6D 69 74 20 45 69 6E 67 61 62 65
[8L7] 85: 20 65 69 6E 65 72 20 5A 61 68 6C 20 64 69 65 20
[2M7] 86: 50 61 72 61 6D 65 74 65 72 20 66 65 73 74 6C 65
[R58] 87: 67 65 6E 20 BA 0A 0D BA 20 2D 2D 2D 2D 2D 2D
[MW7] 88: 2D 2D 2D 2D 2D 2D 2D 2D 50 72 6F 67 72 61 6D 6D

```

```

[6P7] 89: 65 6E 64 65 20 6D 69 74 20 43 74 72 6C 2D 42 72
[728] 90: 65 61 6B 20 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
[HT8] 91: 2D 2D 20 BA 0A 0D BA CD CD CD CD CD CD CD CD
[849] 92: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 93: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 94: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[FL7] 95: CD CD BA 0A 0D BA 20 20 20 20 20 20 20 20 20
[947] 96: 20 20 20 20 20 20 20 20 20 20 20 20 31 20 3D
[C67] 97: 20 20 20 20 20 35 30 20 20 20 20 42 61 75 64 20
[G37] 98: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[5F7] 99: 20 BA 0A 0D BA 20 20 20 20 20 20 20 20 20 20
[M77] 100: 20 20 20 20 20 20 20 20 20 20 32 20 3D 20 20
[R87] 101: 20 20 20 20 37 35 20 20 20 42 61 75 64 20 20
[G37] 102: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[M97] 103: BA 0A 0D BA 20 20 20 20 20 20 20 20 20 20 20
[L67] 104: 20 20 20 20 20 20 20 20 20 20 33 20 3D 20 20
[L67] 105: 20 20 31 31 30 20 20 20 42 61 75 64 20 20 20
[7A7] 106: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 BA
[WA7] 107: 0A 0D BA 20 20 20 20 20 20 20 20 20 20 20 20
[047] 108: 20 20 20 20 20 20 20 20 20 34 20 3D 20 20 20 20
[EB7] 109: 20 31 33 34 2C 35 20 42 61 75 64 20 20 20 20
[FA7] 110: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 BA 0A
[B97] 111: 0D BA 20 20 20 20 20 20 20 20 20 20 20 20 20
[187] 112: 20 20 20 20 20 20 20 20 35 3D 20 20 20 20 20
[R77] 113: 31 35 30 20 20 20 42 61 75 64 20 20 20 20 20
[Z97] 114: 20 20 20 20 20 20 20 20 20 20 20 20 20 BA 0A 0D
[NS7] 115: BA 20 20 41 6B 75 73 74 69 6B 6B 6F 70 70 6C 65
[H97] 116: 72 3A 20 20 20 20 36 20 3D 20 20 20 20 20 33
[M67] 117: 30 30 20 20 20 42 61 75 64 20 20 20 20 20 20
[SK7] 118: 20 20 20 20 20 20 20 20 20 20 20 20 20 BA 0A 0D BA
[G37] 119: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[A97] 120: 20 20 20 20 20 37 3D 20 20 20 20 20 20 20 36 30
[Z67] 121: 30 20 20 20 42 61 75 64 20 20 20 20 20 20 20
[7D7] 122: 20 20 20 20 20 20 20 20 20 20 20 20 BA 0A 0D BA
[G37] 123: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[E87] 124: 20 20 20 20 38 20 3D 20 20 20 20 20 31 32 30 30
[N57] 125: 20 20 20 42 61 75 64 20 20 20 20 20 20 20 20
[EC7] 126: 20 20 20 20 20 20 20 20 20 20 BA 0A 0D BA 20
[G37] 127: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[Y67] 128: 20 20 20 39 20 3D 20 20 20 20 20 31 38 30 30 20
[467] 129: 20 20 42 61 75 64 20 20 20 20 20 20 20 20 20
[5F7] 130: 20 20 20 20 20 20 20 20 20 20 BA 0A 0D BA 20 20
[G37] 131: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[B87] 132: 20 31 30 20 3D 20 20 20 20 20 32 30 30 30 20 20
[377] 133: 20 42 61 75 64 20 20 20 20 20 20 20 20 20 20
[M97] 134: 20 20 20 20 20 20 20 20 20 BA 0A 0D BA 20 20 20
[G37] 135: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[P67] 136: 31 31 20 3D 20 20 20 20 20 32 34 30 30 20 20
[Z67] 137: 42 61 75 64 20 20 20 20 20 20 20 20 20 20 20
[LH7] 138: 20 20 20 20 20 20 20 20 BA 0A 0D BA 20 20 20 20
[X37] 139: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 31
[867] 140: 32 20 3D 20 20 20 20 20 20 33 36 30 30 20 20 42
[Z57] 141: 61 75 64 20 20 20 20 20 20 20 20 20 20 20 20
[AG7] 142: 20 20 20 20 20 20 20 20 BA 0A 0D BA 20 20 20 20
[K47] 143: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 31 33
[3A7] 144: 20 3D 20 20 20 20 20 20 34 38 30 30 20 20 42 61
[357] 145: 75 64 20 20 20 20 20 20 20 20 20 20 20 20 20
[HB7] 146: 20 20 20 20 20 BA 0A 0D BA 20 20 20 20 20 20
[447] 147: 20 20 20 20 20 20 20 20 20 20 20 20 20 31 34 20
[497] 148: 3D 20 20 20 20 20 37 32 30 30 20 20 42 61 75
[W37] 149: 64 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[SK7] 150: 20 20 20 20 BA 0A 0D BA 20 20 20 20 20 20 20
[K87] 151: 20 20 20 20 20 20 20 20 20 20 20 20 31 35 20 3D
[987] 152: 20 20 20 20 20 39 36 30 30 20 20 20 42 61 75 64
[G37] 153: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[R77] 154: 20 20 20 BA 0A 0D BA 20 2D 2D 2D 2D 2D 2D 2D
[CK7] 155: 2D 2D 2D 2D 2D 2D 2D 2D 4E 65 75 65 69 6E 67 61 62
[SG7] 156: 65 20 81 62 65 72 20 44 65 6C 2D 54 61 73 74 65
[858] 157: 20 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
[ET8] 158: 2D 20 BA 0A 0D C8 CD CD CD CD CD CD CD CD CD
[849] 159: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 160: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 161: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[GN7] 162: CD BC 0A 0D 42 69 74 74 65 20 67 65 62 65 6E 20
[HM7] 163: 53 69 65 20 64 69 65 20 67 65 77 81 6E 73 63 68
[E87] 164: 74 65 20 5A 61 68 6C 20 65 69 6E 3A 20 24 20 55
[087] 165: 6E 67 81 6C 74 69 67 65 20 5A 61 68 6C 20 21 20
[PK7] 166: 20 20 4E 65 75 65 69 6E 67 61 62 65 20 81 62 65
[NU7] 167: 72 20 44 65 6C 2D 54 61 73 74 65 20 24 C9 CD CD
[849] 168: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 169: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[019] 170: CD CD CD CD CD CD CD CD CD CD CD CD CD BB 0A 0D BA
[5M7] 171: 20 46 6F 6C 67 65 6E 64 65 20 50 61 72 69 74 84
[AM7] 172: 74 73 81 62 65 72 70 72 81 66 75 6E 67 65 6E 20
[7P7] 173: 73 74 65 68 65 6E 20 62 65 72 65 69 74 20 BA 0A
[G27] 174: 0D BA 20 2D 2D 2D 2D 2D 2D 2D 2D 2D 50 72 6F 67 72
[L77] 175: 61 6D 6D 65 6E 64 65 20 6D 69 74 20 43 74 72 6C
[007] 176: 2D 42 72 65 61 6B 20 2D 2D 2D 2D 2D 2D 2D 2D
[PW8] 177: BA 0A 0D BA CD CD CD CD CD CD CD CD CD CD CD CD
[849] 178: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 179: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[FL7] 180: CD CD BA 0A 0D BA 20 20 20 20 20 20 20 20 20
[647] 181: 20 20 20 20 20 20 20 20 20 20 20 20 30 3D 20 20
[2L7] 182: 20 20 20 4B 65 69 6E 65 20 50 61 72 69 74 84 74
[SK7] 183: 20 20 20 20 BA 0A 0D BA 20 20 20 20 20 20 20 20
[387] 184: 20 20 20 20 20 20 20 20 20 20 20 20 31 20 3D
[6G7] 185: 20 20 55 6E 67 65 72 61 64 65 20 50 61 72 69 74
[OK7] 186: 84 74 20 20 20 20 BA 0A 0D BA 20 20 20 20 20 20
[447] 187: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 32

```



```

[2E7] 188: 20 3D 20 20 20 20 47 65 72 61 64 65 20 50 61 72
[2N7] 189: 69 74 84 74 20 20 20 20 BA 0A 0D BA 20 2D 2D 2D
[DR7] 190: 2D 2D 2D 20 4E 65 75 65 69 6E 67 61 62 65 20 81
[PT7] 191: 62 65 72 20 44 65 6C 2D 54 61 73 74 65 20 2D 2D
[KA8] 192: 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
[849] 193: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 194: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[9V8] 195: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[EK7] 196: 69 74 74 65 20 67 65 62 65 6E 20 53 69 65 20 64
[OP7] 197: 69 65 20 67 65 77 81 6E 73 63 68 74 65 20 5A 61
[MA8] 198: 68 6C 20 65 69 6E 3A 20 24 C9 CD CD CD CD CD CD
[849] 199: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 200: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[W28] 201: CD BB 0A 0D BA 20 44 69 65 20 66 6F 6C 67 65 6E
[SH7] 202: 64 65 6E 20 44 61 74 65 6E 62 69 74 73 20 73 74
[XX7] 203: 65 68 65 6E 20 62 65 72 65 69 74 20 BA 0A 0D BA
[TX7] 204: 20 2D 2D 2D 2D 20 50 72 6F 67 72 61 6D 6D 65 6E
[KH7] 205: 64 65 20 6D 69 74 20 43 74 72 6C 2D 42 72 65 61
[RE8] 206: 68 20 2D 2D 2D 2D 20 BA 0A 0D BA CD CD CD CD CD
[849] 207: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 208: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[FL7] 209: CD CD BA 0A 0D BA 20 20 20 20 20 20 20 20 20 20
[277] 210: 20 20 20 20 20 20 20 20 20 20 35 20 3D 20 35 20
[1N7] 211: 44 61 74 65 6E 62 69 74 73 20 20 20 20 BA 0A 0D
[257] 212: BA 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[7G7] 213: 20 20 20 20 36 20 3D 20 20 36 20 44 61 74 65 6E
[FD7] 214: 62 69 74 73 20 20 20 20 BA 0A 0D BA 20 20 20 20
[757] 215: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 37
[5G7] 216: 20 3D 20 20 37 20 44 61 74 65 6E 62 69 74 73 20
[7D7] 217: 20 20 20 BA 0A 0D BA 20 20 20 20 20 20 20 20 20
[9K7] 218: 20 20 20 20 20 20 20 20 20 20 38 20 3D 20 20 38
[2L7] 219: 20 44 61 74 65 6E 62 69 74 73 20 20 20 BA 0A 0D
[T28] 220: 0D BA 20 2D 2D 2D 2D 2D 2D 2D 2D 4E 65 75 65 69 6E
[PK7] 221: 61 62 65 20 81 62 65 72 20 44 65 6C 2D 54 61 73
[R88] 222: 74 65 20 2D 2D 2D 2D 2D 2D BA 0A 0D CD CD CD CD
[849] 223: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 224: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[218] 225: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[2P7] 226: 65 6E 20 53 69 65 20 64 69 65 20 67 65 77 81 6E
[AN7] 227: 73 63 68 74 65 20 5A 61 68 6C 20 65 69 6E 3A 20
[019] 228: 24 C9 CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 229: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[WM8] 230: CD CD CD CD CD CD CD CD CD CD CD BB 0A 0D BA 20 44 69
[MT7] 231: 65 20 66 6F 6C 67 65 6E 64 65 6E 20 53 74 6F 70
[SH7] 232: 70 62 69 74 73 20 73 74 65 68 65 6E 20 62 65 72
[L08] 233: 65 69 74 20 BA 0A 0D BA 20 2D 2D 2D 2D 20 50 72
[MU7] 234: 6F 67 72 61 6D 6D 65 6E 64 65 20 6D 69 74 20 43
[HS7] 235: 74 72 6C 2D 42 72 65 61 6B 20 2D 2D 2D 2D BA
[W09] 236: 0A 0D BA CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 237: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[VL8] 238: CD CD CD CD CD CD CD CD CD CD CD BA 0A 0D BA 20 20
[947] 239: 20 20 20 20 20 20 20 20 20 20 31 20 3D 20 20
[9D7] 240: 31 20 53 74 6F 70 70 62 69 74 20 20 20 20 20 20
[HB7] 241: 20 20 20 20 20 BA 0A 0D BA 20 20 20 20 20 20 20
[OC7] 242: 20 20 20 20 20 20 32 20 3D 20 32 20 53 74 6F
[T77] 243: 70 70 62 69 74 20 20 20 20 20 20 20 20 20 20 20
[1X7] 244: BA 0A 0D BA 20 2D 2D 2D 2D 2D 2D 4E 65 75 65 69
[8L7] 245: 6E 67 61 62 65 20 81 62 65 72 20 44 65 6C 2D 54
[R28] 246: 61 73 74 65 20 2D 2D 2D 2D 2D 2D BA 0A 0D CD CD
[849] 247: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 248: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 249: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[PH7] 250: 65 62 65 6E 20 53 69 65 20 64 69 65 20 67 65 77
[4S7] 251: 81 6E 73 63 68 74 65 20 5A 61 68 6C 20 65 69 6E
[AH8] 252: 3A 20 24 0A 0A 0A 0D C9 CD CD CD CD CD CD CD CD
[849] 253: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 254: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 255: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[M39] 256: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD BB
[WS7] 257: 0A 0D BA 20 20 20 41 75 66 20 64 65 6D 20 61 6B
[1V7] 258: 74 75 65 6C 6C 65 6E 20 4C 61 75 66 77 65 72 68
[NK7] 259: 20 28 64 61 73 20 4C 61 75 66 77 65 72 6B 2C 20
[7L7] 260: 76 6F 6E 20 64 65 6D 20 64 69 65 73 65 73 20 50
[BP7] 261: 72 6F 67 72 61 6D 6D 20 20 20 BA 0A 0D BA 20 20
[BH7] 262: 67 65 73 74 61 72 74 65 74 20 77 75 72 64 65 29
[UM7] 263: 20 77 69 72 64 20 65 69 6E 20 44 46 9A 2D 50 72
[GT7] 264: 6F 74 6F 6B 6F 6C 6C 20 67 65 73 70 65 69 63 68
[PK7] 265: 65 72 74 2E 20 4E 61 63 68 64 65 6D 20 53 69 65
[2M7] 266: 20 64 61 73 20 BA 0A 0D BA 20 20 20 50 72 6F 67
[DG7] 267: 72 61 6D 6D 20 62 65 65 6E 64 65 74 20 68 61 62
[2P7] 268: 65 6E 2C 20 6C 61 73 73 65 6E 20 73 69 63 68 20
[7K7] 269: 64 69 65 20 44 61 74 65 6E 20 6D 69 74 20 64 65
[WK7] 270: 6D 20 54 79 70 65 2D 42 65 66 65 68 6C 20 20 20
[1V7] 271: BA 0A 0D BA 20 61 75 73 64 72 75 63 68 65 6E 2E
[ME7] 272: 20 47 65 62 65 6E 20 53 69 65 20 64 61 7A 75 20
[V08] 273: 6E 75 72 20 AF 74 79 70 65 AE 20 67 65 66 6F 6C
[EP7] 274: 67 74 20 76 6F 6E 20 64 65 6D 20 44 61 74 65 69
[F58] 275: 6E 61 6D 65 6E 20 65 69 6E 2E 20 BA 0A 0D BA CD
[849] 276: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 277: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 278: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 279: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[TD8] 280: CD CD CD CD CD CD BA 0A 0D BA 20 47 65 62 65 6E
[YM7] 281: 20 53 69 65 20 6E 75 6E 20 64 65 6E 20 67 65 77
[OU7] 282: 81 6E 73 63 68 74 65 6E 20 44 61 74 65 69 6E 61
[5K7] 283: 6D 65 6E 20 66 81 72 20 64 69 65 20 50 72 6F 74
[GU7] 284: 6F 6B 6F 6C 6C 64 61 74 65 69 20 65 69 6E 2E 20
[YP7] 285: 20 BA 0A 0D BA 20 20 44 65 72 20 44 61 74 65 69

```

```

[4T7] 286: 6E 61 6D 65 6E 20 64 61 72 66 20 6D 61 78 69 6D
[1L7] 287: 61 6C 20 61 63 68 74 20 5A 65 69 63 68 65 6E 20
[7P7] 288: 6C 61 6E 67 20 73 65 69 6E 20 75 6E 64 20 6E 75
[6U7] 289: 72 20 61 75 73 20 64 65 6E 20 20 20 BA 0A 0D BA
[3G7] 290: 20 42 75 63 68 73 74 61 62 65 6E 20 41 20 62 69
[DH7] 291: 73 20 5A 20 75 6E 64 20 64 65 6E 20 5A 69 66 66
[MD7] 292: 65 72 6E 20 30 20 62 69 73 20 39 20 62 65 73 74
[2R7] 293: 65 68 65 6E 2E 20 44 69 65 20 45 72 77 65 69 74
[6Y7] 294: 65 72 75 6E 67 20 20 BA 0A 0D BA 20 20 20 AF 74
[DP7] 295: 78 74 AE 20 66 81 67 74 20 64 61 73 20 50 72 6F
[HS7] 296: 67 72 61 6D 6D 20 61 75 74 6F 6D 61 74 69 73 63
[RN7] 297: 68 20 61 6E 2E 20 42 65 69 20 46 65 68 6C 65 69
[9H7] 298: 6E 67 61 62 65 6E 20 77 69 72 64 20 64 65 72 20
[EC7] 299: 20 20 BA 0A 0D BA 20 20 20 20 20 20 20 20 20 20
[D77] 300: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[BP7] 301: 69 6E 61 6D 65 20 65 72 6E 65 75 74 20 61 62 67
[GB7] 302: 65 66 72 61 67 74 2E 20 65 72 65 20 20 20 20 20
[297] 303: 20 20 20 20 20 20 20 20 20 20 20 20 20 BA 0A 0D
[G39] 304: C8 CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 305: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 306: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 307: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[2C8] 308: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[2U7] 309: 64 65 72 20 50 72 6F 74 6F 6B 6F 6C 64 61 74
[PA8] 310: 65 69 3A 20 24 0A 0A 0A 0D C9 CD CD CD CD CD CD
[849] 311: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 312: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 313: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 314: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[FT7] 315: CD BB 0A 0D BA 20 20 44 69 65 20 44 61 74 65 69
[BK7] 316: 20 69 73 74 20 62 65 72 65 69 74 73 20 76 6F 72
[TK7] 317: 68 61 6E 64 65 6E 2E 20 9A 62 65 72 73 63 68 72
[VP7] 318: 65 69 62 65 6E 20 74 72 6F 74 7A 64 65 6D 20 65
[PY7] 319: 72 77 81 6E 73 63 68 74 20 3F 20 20 BA 0A 0D C8
[849] 320: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 321: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 322: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 323: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[KB8] 324: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[4T7] 325: 69 6E 67 61 62 65 20 BC 0A 2F 4E 29 3A 20 24 0A
[EV8] 326: 0A 0A 0D C9 CD CD CD CD CD CD CD CD CD CD CD CD
[849] 327: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 328: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[5C8] 329: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[7V7] 330: 65 73 65 20 4B 6F 6E 66 69 67 75 72 61 74 69 6F
[YG7] 331: 6E 20 61 62 67 65 73 70 65 69 63 68 65 72 74 20
[648] 332: 77 65 72 64 65 6E 3F 20 BA 0A 0D CD CD CD CD
[849] 333: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 334: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[UW8] 335: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[CS7] 336: 68 72 65 20 45 69 6E 67 61 62 65 20 28 4A 2F 4E
[FK8] 337: 29 3A 20 24 0A 0A 0D C9 CD CD CD CD CD CD CD CD
[849] 338: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[849] 339: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[L28] 340: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[KS7] 341: 69 6E 64 20 66 6F 6C 67 65 6E 64 65 20 44 61 74
[ZY7] 342: 65 6E 20 76 6F 72 65 69 6E 67 65 73 74 65 6C 6C
[MP7] 343: 74 20 3A 20 20 BA 0A 0D 24 0A 0A 0A 0A 0A 0A
[F67] 344: 0A 0A 0A 20 20 20 20 20 20 20 20 20 20 20 20
[G37] 345: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[S09] 346: 20 C9 CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[XT7] 347: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[G37] 348: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[WN7] 349: 20 20 20 20 20 BA 20 4D 6F 6D 65 6E 74 20 62 69
[TG7] 350: 74 74 65 20 2E 2E 2E 20 BA 0A 0D 20 20 20 20 20
[G37] 351: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[708] 352: 20 20 20 20 20 20 20 20 20 20 C8 CD CD CD CD CD
[ZR8] 353: CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
[HP7] 354: 4D 69 6E 75 74 65 20 57 61 72 74 65 74 65 69 74
[GP7] 355: 2C 20 69 6E 6E 65 72 68 61 6C 62 20 64 65 72 20
[3L7] 356: 73 69 63 68 20 64 69 65 20 47 65 67 65 6E 73 74
[VK7] 357: 65 6C 6C 65 20 6D 69 74 20 43 61 72 72 69 65 72
[NU7] 358: 20 6D 65 6C 64 65 6E 20 6D 75 E1 2E 24 44 61 74
[NT7] 359: 65 6E 81 62 65 72 74 72 61 67 75 6E 67 20 61 6B
[LG7] 360: 74 69 76 2C 20 41 62 62 72 75 63 68 20 81 62 65
[9T7] 361: 72 20 64 69 65 20 46 75 6E 6B 74 69 6F 6E 73 74
[1C7] 362: 61 73 74 65 20 5B 46 31 30 5D 20 20 20 20 20 20
[T77] 363: 20 20 20 20 20 20 20 20 20 20 20 20 24 44 61 74 65
[DK7] 364: 6E 81 62 65 72 74 72 61 67 75 6E 67 20 69 6E 61
[S77] 365: 6B 74 69 76 20 20 20 20 20 20 20 20 20 20 20 20
[G37] 366: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[G37] 367: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[T47] 368: 20 20 20 20 20 20 20 20 20 20 20 20 24 07 24 20 20
[G37] 369: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
: : : : : : : : : : : : : : : :
[G37] 1043: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[557] 1044: 20 20 00 20 20 20 20 20 20 20 20 20 54 58 54 00 00
[007] 1045: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[X37] 1046: 00 00 00 00 00 00 00 00 00 00 44 46 55 20 20 20 20
[267] 1047: 4B 4F 4E 00 00 00 00 00 00 00 00 00 00 00 00 00
[937] 1048: 00 00 00 00 00 00 00 00 00 00 00 00 00 44 46 55
[7A7] 1049: 20 20 20 20 20 20 4B 4F 4E 00 00 00 00 00 00 00
[007] 1050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 1051: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[DW7] 1052: FB 1E B8 00 00 50 B8 04 00 8E D8 E8 14 08 BA 00
[6W7] 1053: 00 B4 02 CD 10 E8 A2 07 3C FF 74 20 B4 00 CD 16
[TX7] 1054: 3C 6A 74 15 3C 4A 74 11 3C 6E 74 10 3C 4E 74 0C
[F38] 1055: BA CB 14 B4 09 CD 21 EB E3 E9 47 04 BA FB 03 B0

```



```
[308] 1056: 80 EE E8 DD 07 BA 00 00 B4 02 CD 10 BA 7A 02 B4
[318] 1057: 09 CD 21 BF CD 14 B3 00 B4 01 CD 21 3C 08 74 E2
[318] 1058: 3C 39 77 1D 3C 0D 74 1E 3C 30 72 15 88 05 47 FE
[318] 1059: C3 80 FB 03 74 CC B4 03 CD 10 80 FA 2C 73 02 EB
[398] 1060: D7 E8 16 06 EB CD 80 FB 00 74 B7 80 FB 02 74 5D
[398] 1061: BE CD 14 8A 04 2C 30 46 B4 0D 88 24 BA F8 03 3C
[398] 1062: 00 74 9F 3C 01 74 22 3C 02 74 23 3C 03 74 24 3C
[398] 1063: 04 74 25 3C 05 74 26 3C 06 74 27 3C 07 74 28 3C
[398] 1064: 08 74 29 3C 08 74 2A 3C 09 74 2B 3C 0A 74 2C 3C
[398] 1065: 09 74 2D 3C 09 74 2E 3C 0A 74 2F 3C 0B 74 30 3C
[398] 1066: A9 00 B0 80 E9 AE 00 E8 B0 05 E9 66 FF BE CD 14
[398] 1067: 8A 24 80 EC 30 80 FC 02 73 ED 46 8A 04 2C 30 3C
[398] 1068: 05 77 E4 02 C4 3C 07 73 DE 46 B4 0D 88 24 BA F8
[398] 1069: 03 3C 01 74 26 3C 02 74 27 3C 03 74 28 3C 04 74
[398] 1070: 29 3C 05 74 2A 3C 06 74 2B 3C 07 74 2C 3C 08 74
[398] 1071: 90 B0 60 EB 24 90 B0 40 EB 1F 90 B0 3A EB 1A 90
[398] 1072: B0 30 EB 15 90 B0 20 EB 10 90 B0 18 EB 0B 90 B0
[398] 1073: 10 EB 06 90 B0 0C EB 01 90 EE BA F9 03 B0 00 EE
[398] 1074: EB 3D 90 EE BA F9 03 B0 09 EE BA F9 03 B0 00 EE
[398] 1075: 03 B0 06 EE BA F9 03 B0 09 EE BA F9 03 B0 00 EE
[398] 1076: 90 EE BA F9 03 B0 09 EE BA F9 03 B0 00 EE
[398] 1077: 03 EE EB 0B 90 EE BA F9 03 B0 01 EE EB 01 90 BA
[398] 1078: FB 03 B0 00 EE EB 7A 06 BA 00 00 B4 02 CD 10 BA
[398] 1079: 3D 08 B4 09 CD 21 BF D7 14 B3 00 B4 01 CD 21 3C
[398] 1080: 08 74 DC 3C 32 77 18 3C 0D 74 19 3C 02 72 10 88
[398] 1081: 05 47 FE C3 B4 03 CD 10 80 FA 2B 73 C2 EB DC E8
[398] 1082: B8 04 EB D8 80 FB 00 74 B6 BE D7 14 8A 04 2C 30
[398] 1083: BA FB 03 3C 00 74 0A 3C 01 74 0B 3C 02 74 0C EB
[398] 1084: 9E B0 00 EB 0B 90 B0 10 EB 06 90 B0 00 EB 01 90
[398] 1085: EE E8 0E 06 BA 00 00 B4 02 CD 10 BA 29 0A B4 09
[398] 1086: CD 21 BF E1 14 B3 00 B4 01 CD 21 3C 08 74 E2 3C
[398] 1087: 38 77 18 3C 0D 74 19 3C 35 72 10 88 05 47 FE C3
[398] 1088: B4 03 CD 10 80 FA 2B 73 C8 EB DC E8 4C 0A EB D2
[398] 1089: 80 FB 00 74 BC BE E1 14 8A 04 2C 30 BA FB 03 3C
[398] 1090: 05 74 0E 3C 06 74 0F 3C 07 74 10 3C 08 74 11 EB
[398] 1091: A0 B4 00 EB 10 90 B4 01 EE 0B 90 B4 02 EB 06 90
[398] 1092: B4 03 EB 01 90 EC 0A C4 EE E8 96 05 BA 00 00 B4
[398] 1093: 02 CD 10 BA 01 0C B4 09 CD 21 BF EB 14 B3 00 B4
[398] 1094: 01 CD 21 3C 08 74 E2 3C 32 77 18 3C 0D 74 19 3C
[398] 1095: 31 72 10 88 05 47 FE C3 B4 03 CD 10 80 FA 2B 73
[398] 1096: C8 EB DC E8 4C 0A EB D2 80 FB 00 74 BC BE EB 14
[398] 1097: 8A 04 2C 30 BA FB 03 3C 01 74 06 3C 02 74 07 EB
[398] 1098: A8 B4 00 EB 06 90 B4 04 EB 01 90 EC 0A C4 EE E8
[398] 1099: 30 05 BA 00 00 B4 02 CD 10 BA 37 01 B4 09 CD 21
[398] 1100: BE CD 14 46 8A 04 3C 0D 74 07 46 8A 04 3C 0D 74
[398] 1101: 40 BE CD 14 8A 04 2C 30 3C 01 73 06 3C 09 76 02
[398] 1102: EB CD B4 03 CD 10 52 B2 0C B6 01 B4 02 CD 10 BA
[398] 1103: 00 00 3C 01 74 0D B3 0B F6 E3 3C 01 74 05 42 FE
[398] 1104: C8 EB F7 B4 09 CD 21 5A FE C2 B4 02 CD 10 EB 46
[398] 1105: 90 BE CD 14 8A 24 80 EC 30 46 8A 04 2C 30 02 C4
[398] 1106: 3C 01 73 06 3C 06 76 02 EB 85 B4 03 CD 10 52 B2
[398] 1107: 0C B6 01 B4 02 CD 10 BA 6B 00 3C 01 74 0D B3 0B
[398] 1108: F6 E3 3C 01 74 05 42 FE C8 EB F7 B4 09 CD 21 5A
[398] 1109: FE C2 B4 02 CD 10 BE D7 14 8A 04 2C 30 3C 00 73
[398] 1110: 06 3C 02 76 02 EB EF B4 03 CD 10 52 B2 0C B6 02
[398] 1111: B4 02 CD 10 BA B5 00 3C 00 74 0D B3 0A F6 E3 3C
[398] 1112: 00 74 05 42 FE C8 EB F7 B4 09 CD 21 5A B4 02 CD
[398] 1113: 10 BE E1 14 8A 04 2C 30 3C 05 73 06 3C 08 76 02
[398] 1114: EB EF B4 03 CD 10 52 B2 0C B6 03 B4 02 CD 10 BA
[398] 1115: D2 00 3C 05 74 0D B3 0B F6 E3 3C 00 74 05 42 FE
[398] 1116: C8 EB F7 B4 09 CD 21 5A B4 02 CD 10 BE EB 14 8A
[398] 1117: 04 2C 30 3C 01 73 06 3C 02 76 02 EB EF B4 03 CD
[398] 1118: 10 52 B2 13 B6 04 B4 02 CD 10 BA 33 01 3C 01 74
[398] 1119: 02 42 42 B4 09 CD 21 5A B4 02 CD 10 B4 00 CD 16
[398] 1120: 3C 6A 74 18 3C 4A 74 14 3C 6E 74 0D 3C 4E 74 09
[398] 1121: BA CB 14 B4 09 CD 21 EB E3 E9 E0 FB BA 1F 12 B4
[398] 1122: 09 CD 21 B4 0D CD 16 3C 6A 74 15 3C 4A 74 11 3C
[398] 1123: 6E 74 10 3C 4E 74 0C BA CB 14 B4 09 CD 21 EB E3
```

```
[NS7] 1124: E8 AC 02 E8 9C 03 B1 00 BF 02 3F 47 B0 20 88 05
[R88] 1125: 47 FE C1 80 F9 08 74 02 EB F2 BA 83 0D B4 09 CD
[HX7] 1126: 21 B1 00 BF 02 3F 47 B4 00 CD 16 3C 0D 74 17 3C
[X38] 1127: 30 72 D0 3C 7A 77 CC B4 0E CD 10 80 F9 08 74 06
[F48] 1128: FE C1 88 05 EB E0 80 F9 00 74 B8 BA 02 3F B4 0F
[4P7] 1129: CD 21 3C FF 74 31 BA 02 3F B4 10 CD 21 E8 42 03
[0X7] 1130: BA 25 11 B4 09 CD 21 B4 00 CD 16 3C 6A 74 18 3C
[C18] 1131: 4A 74 14 3C 6E 74 0D 3C 4E 74 09 BA CB 14 B4 09
[848] 1132: CD 21 EB E3 E9 7C FF E8 18 03 B2 00 B6 18 B4 02
[508] 1133: CD 10 BA DE 13 B4 09 CD 21 BA 00 00 B4 02 CD 10
[BG8] 1134: BB FF FF B1 3C BA FC 03 B0 01 EE BA FE 03 EC A8
[MZ7] 1135: 20 75 1B A8 80 75 17 4B 83 FB 00 74 02 EB EC FE
[YT7] 1136: C9 80 F9 00 74 02 EB E3 E8 20 01 E9 13 01 BB 00
[R38] 1137: 00 E8 CE 02 E8 F9 02 BA 00 00 B4 02 CD 10 BA 02
[CY7] 1138: 17 B4 1A CD 21 BF 02 17 B9 00 00 BA FE 03 EC A8
[488] 1139: 80 74 5F BA FD 03 EC A8 01 74 5A BA F8 03 EC 88
[VR7] 1140: 05 47 41 81 F9 00 28 74 46 3C 0D 74 1B 3C 08 74
[KR7] 1141: 11 B4 0E CD 10 51 B4 03 CD 10 59 80 FE 18 74 19
[678] 1142: EB C9 4F 4F 49 49 EB E9 B4 0E CD 10 51 B4 03 CD
[408] 1143: 10 59 80 FE 17 73 02 EB B2 52 E8 7D 02 B0 0A B4
[KC8] 1144: 0E CD 10 E8 8A 02 5A FE CE B4 02 CD 10 EB 9C EB
[P38] 1145: 4E 90 EB 3E 90 E8 4E 01 B4 0B CD 21 3C 00 74 ED
[MY7] 1146: B4 00 CD 16 80 FC 44 74 E9 80 FE 43 74 0F 3C 0D
[BA8] 1147: 74 0F 80 FC 4B 74 17 BA F8 03 EC D0 B0 18 EB
[2A8] 1148: F6 BA F8 03 B0 0D EE E8 1C 01 B0 0A EB E9 B0 08
[528] 1149: EB E5 E8 11 01 B0 18 BA F8 03 EE B0 1A 88 05 B4
[3Y7] 1150: 00 50 E8 01 01 B0 13 BA F8 03 EE BA 02 3F B4 0F
[DV7] 1151: CD 21 3C FF 74 22 BA 02 3F B9 50 00 B4 28 CD 21
[CP7] 1152: BA 02 3F B4 10 CD 21 58 3C 1A 74 15 E8 7D 00 B0
[F98] 1153: 11 BA F8 03 EE E9 06 FF BA 02 3F B4 16 CD 21 EB
[8X7] 1154: CA E8 07 00 E8 D3 01 E8 B8 01 CB B6 18 B2 00 B4
[GY7] 1155: 02 CD 10 BA 7C 14 B4 09 CD 21 E8 A9 00 BA F8 03
[TC8] 1156: B0 18 EE B1 0A BB FF FF 4B 83 FB 04 02 EB F8
[Y38] 1157: FE C9 80 F9 00 74 02 EB EC 3C B4 03 CD 10 52 53
[5X7] 1158: BA CB 14 B4 09 CD 21 B2 00 B6 18 B4 02 CD 10 BF
[FP7] 1159: 0E 08 8A 05 3C 24 74 12 47 B9 01 00 B3 70 B4 09
[PA8] 1160: CD 10 FE C2 B4 02 CD 10 EB E8 B0 05 BB FF FF 4B
[028] 1161: 83 FB 00 74 02 EB F8 FE C8 3C 00 74 02 EB ED B2
[4N7] 1162: 00 B6 18 B4 02 CD 10 B0 20 B9 2E 00 B3 02 B4 09
[GX7] 1163: CD 10 5B 5A FE CA 80 FB 02 74 13 B4 02 CD 10 B0
[PL7] 1164: 20 B4 0A B9 01 00 CD 10 80 FB 02 74 01 C3 B2 15
[UK7] 1165: B6 16 52 B4 02 CD 10 B0 20 B4 0A B9 02 00 CD 10
[P88] 1166: 5A B4 02 CD 10 C3 BA FD 03 EC A8 20 74 F8 C3 BA
[RE8] 1167: FF 14 B4 1A CD 21 BF FF 14 BA FB 03 EC B4 00 50
[728] 1168: BA FB 03 B0 80 EE BA F8 03 EC 88 05 47 BA F9 03
[L48] 1169: EC 88 05 47 BA FB 03 B0 00 EE 58 88 05 47 BA FB
[2Y7] 1170: 03 EE BA 00 00 B4 02 CD 10 B4 08 CD 10 88 05 3C
[M28] 1171: 4E 74 18 80 FA 50 74 09 47 FE C2 B4 02 CD 10 EB
[D48] 1172: E8 B2 00 FE C6 B4 02 CD 10 EB DE 47 B0 1A 88 05
[HW7] 1173: E8 8F 00 BA 00 00 B4 02 CD 10 BA 3A 13 B4 09 CD
[0V7] 1174: 21 BA 27 3F B4 16 CD 21 BA 27 3F B9 04 00 B4 28
[898] 1175: CD 21 BA 27 3F B4 10 CD 21 C3 BA FF 14 B4 1A CD
[V38] 1176: 21 BA 4C 3F B4 0F CD 21 3C 00 75 55 BA 4C 3F B9
[438] 1177: 04 00 B4 27 CD 21 BA 4C 3F B4 10 CD 21 BF FF 14
[D38] 1178: BA FB 03 B0 80 EE 8A 05 47 BA F8 03 EE 8A 05 47
[Y28] 1179: BA F9 03 EE BA FB 03 B0 00 EE 8A 05 47 BA FB 03
[H18] 1180: EE BA D4 12 B4 09 CD 21 B1 50 47 FE C9 80 F9 00
[L28] 1181: 74 02 EB F6 8A 05 B4 0E CD 10 3C 4E 74 03 47 EB
[2V7] 1182: F3 C3 BA 00 00 B4 02 CD 10 B0 20 B4 0A B9 0D 07
[1T7] 1183: CD 10 BA 00 00 B4 02 CD 10 C3 51 B2 00 B6 18 B4
[727] 1184: 02 CD 10 B0 20 B4 09 B3 0F B9 50 00 CD 10 59 C3
[AT7] 1185: B2 00 B6 18 B4 02 CD 10 BE 2D 14 8A 04 3C 24 74
[2W7] 1186: 14 46 51 B9 01 00 B3 70 B4 09 CD 10 FE C2 59 B4
[BS2] 1187: 02 CD 10 EB E6 C3
```

Listing 2. Wenn Sie keinen Assembler besitzen, können Sie »STrans« auch als Hexadezimal-Listing eingeben und mit »SCheck« übersetzen

Lieben Sie Parties? – Wir planen bereits Ihre Einstandsfeier

Wenn Sie Interesse haben, in einem jungen unkonventionellen Team in der Computerbranche mitzuarbeiten, dann sind Sie bei uns richtig. Wir suchen Fachredakteure mit Berufserfahrung für die Aufgabengebiete

- * Software-Anwendungen
- * Programmiersprachen
- * Betriebssysteme

Wir bieten Ihnen Freiräume für die Umsetzung eigener Ideen. Eine leistungsorientierte Vergütung, flexible Arbeitszeit und eine zusätzliche Altersversorgung sind bei uns selbstverständlich.

Von Ihnen erwarten wir Einsatzbereitschaft und Eigeninitiative. Bitte richten Sie Ihre Bewerbung mit den üblichen Unterlagen und einigen Arbeitsproben an

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

Wie »STrans« funktioniert

Das Herz der seriellen Schnittstelle wird in den meisten PCs von dem integrierten Baustein 8250 gebildet. Das DFÜ-Programm »STrans« programmiert diesen Baustein auf direktem Weg in Assembler. Wie das im einzelnen geschieht, erklärt dieser Beitrag.

Die Grundlage des DFÜ-Programms »STrans«, dessen Quellcode in dieser Ausgabe der DOS Extra abgedruckt ist, bildet die Programmierung des Schnittstellen-Bausteins 8250. Dieser Baustein ist in den meisten PCs enthalten, wird bei einigen Modellen jedoch durch den Typ 8251 ersetzt.

Der Baustein 8250 verfügt über verschiedene Register, die eine Datenübertragung steuern. Diese Register, die auch im Quelltext von STrans aufgeführt sind, listet Tabelle 1 nochmals auf.

Damit sich für den 8250-Baustein eine bestimmte Baudrate programmieren läßt, muß Bit 7 im Leitungssteuerregister auf 1 gesetzt sein. Dieser Zustand zeigt an, daß die beiden Register unter den Portadressen 3F8 und 3F9hex gegenwärtig als Baudratenregister benutzt werden. Das Bit 7 wird gesetzt, indem der hexadezimale Wert 80 in das Leitungssteuerregister geschrieben wird. Ein Beispiel:

```
MOV DX,3FBH ;Leitungssteuerregister
               vorbelegen
MOV AL,80H   ;Wert für Bit 7=1 in AL über-
               tragen
OUT DX,AL    ;Bit 7 im Leitungssteuerre-
               gister setzen
```

Jetzt läßt sich ein Wert für die gewünschte Baudrate in das Baudratenregister schreiben. Tabelle 2 zeigt, welche Werte welche Baudraten bewirken. Wie Sie sehen, ist es leider nicht möglich, einfach die Höhe der Baudrate in das Register zu schreiben. Vielmehr gibt der in das Baudratenregister geschriebene Wert nach der Formel

$$115200 / \text{Wert} = \text{Baudrate}$$

die Höhe der Baudrate an.

Das folgende Beispiel legt die Baudrate auf 300 Baud fest:

```
MOV DX,3F8H ;Baudratenregister (MSB)
               vorbelegen
MOV AL,80H  ;Wert für 300 Baud (MSB)
OUT DX,AL   ;setze höherwertiges Byte
```

```
MOV DX,3F9H ;Baudratenregister (LSB)
               vorbelegen
MOV AL,01H  ;Wert für 300 Baud (LSB)
OUT DX,AL   ;setze niederwertiges Byte
```

Ab einer Baudrate von 600 Baud ist das höherwertige Byte des Baudratenregisters durchgehend bis 9600 Baud mit Null zu programmieren, weil der Teiler für den Ausgangswert von 115200 sehr klein wird und sich dementsprechend nur das niederwertige Byte ändert.

Um eine Parität für die Datenübertragung zu programmieren, muß das

Bit 4 im Leitungssteuerregister auf den entsprechenden Wert gesetzt werden. Ist das Bit auf 1 gesetzt, so arbeitet der 8250-Baustein mit gerader Parität, andernfalls mit ungerader beziehungsweise keiner Parität.

Die Parität wird mit folgendem Beispiel auf »gerade« gesetzt:

```
MOV DX,3FBH ;Leitungssteuerregister
               vorbelegen
MOV AL,10H  ;Wert für Bit 4=1 in AL über-
               tragen
OUT DX,AL   ;Bit 4 im Leitungssteuerre-
               gister setzen
```

Darauf sollte die Anzahl der zu übertragenen Datenbits pro Bitpaket defi-

niert werden. Dies geschieht ebenfalls über das Leitungssteuerregister. Um die Einstellung von Bit 4 für die Parität jedoch nicht zu überschreiben, wählen Sie anstelle des MOV-Befehls den OR-Befehl für eine Oder-Verknüpfung.

Die Parameter für die Datenübertragung werden im Leitungssteuerregister des 8250-Bausteins eingestellt

Bit 0 und Bit 1 geben die Anzahl der Datenbits an:

Bit 1 = 0 und Bit 0 = 0: 5 Datenbit
 Bit 1 = 0 und Bit 0 = 1: 6 Datenbit
 Bit 1 = 1 und Bit 0 = 0: 7 Datenbit
 Bit 1 = 1 und Bit 0 = 1: 8 Datenbit

Um beispielsweise einen Wert von acht Datenbit pro Bitpaket einzustellen, ist folgende Befehlsfolge notwendig:

Portadresse	Register
3F8hex	Sende-Empfangspuffer und Baudratenregister (höherwertiges Byte)
3F9hex	Unterbrechung-Aktivierungsregister und Baudratenregister (niederwertiges Byte)
3FAhex	Unterbrechung-Erkennungsregister
3FBhex	Leitungssteuerregister
3FChex	Modemsteuerregister
3FDhex	Leitungssteuerregister
3FEhex	Modemstatusregister

Tabelle 1. Die Register des Schnittstellen-Bausteins 8250 mit den zugehörigen Portadressen

Baudrate	höherwertiges Byte (LSB)	niederwertiges Byte (MSB)
50	09	00
75	06	00
110	04	17
134,5	03	59
150	03	00
300	01	80
600	00	C0
1200	00	60
1800	00	40
2000	00	3A
2400	00	30
3600	00	20
4800	00	18
7200	00	10
9600	00	0C

(Alle Byte-Werte in hexadezimaler Notation)

Tabelle 2. Über die beiden Baudratenregister wird die Geschwindigkeit für die Datenübertragung eingestellt

Bit	Signal	Bedeutung
Bit 4 = 0	Clear To Send	Kein Datenton
Bit 4 = 1	Clear To Send	Datenton wird empfangen
Bit 5 = 0	Data Set Ready	Modem ist nicht betriebsbereit
Bit 5 = 1	Data Set Ready	Modem ist betriebsbereit
Bit 7 = 0	Data Carrier Detect	Keine Verbindung zum Partnermodem
Bit 7 = 1	Data Carrier Detect	Verbindung zum Partnermodem

Tabelle 3. Die Bit 4, 5 und 7 des Modemstatusregisters zeigen wichtige Informationen an


```
MOV DX,3FBH ;Leitungssteuerregister
               vorbelegen
MOV AH,03H   ;Bit 1 und 0 =1 für 8 Datenbit
IN AL,DX     ;vorbelegten Wert aus DX in
               AL einlesen
OR AL,AH     ;AL mit AH über Oder-Funktion
               verknüpfen
OUT DX,AL    ;und ausgeben, alle Bits
               bleiben gesetzt
```

```
OUT DX,AL    ;und ausgeben, alle Bits
               bleiben gesetzt
```

Um jetzt mit dem 8250 Datenfernübertragung betreiben zu können, benötigt man allerdings noch mehr Wissen um das Innenleben dieses Bausteins.

So ist beispielsweise im Modemsteuerregister das Bit 0 auf 1 zu setzen, um anzuzeigen, daß das Datenendgerät für die Datenübertragung bereit ist:

```
MOV DX,3FCH ;Modemsteuerregister vorbelegen
MOV AL,01H  ;Bit 0=1 für Datenendgerät
               bereit
OUT DX,AL   ;Bit 0 im Modemsteuerregister
               setzen
```

Ansonsten ist das Modemsteuerregister allerdings relativ unwichtig.

Von größerem Interesse ist da schon das Modemstatusregister, denn mit diesem Register läßt sich überprüfen, ob ein Trägersignal (Carrier) von der Gegenstelle empfangen wird, ob das Modem noch zum Senden bereit ist, oder ob die Gegenstelle noch sendet.

Tabelle 3 zeigt die Bedeutung der Bit 4, 5 und 7 des Modemstatusregisters.

Im folgenden ein praktisches Beispiel, das durch Abfrage von Bit 5 überprüft, ob sich die Gegenstelle meldet:

```
MOV DX,3FEH ;Modemstatusregister vorbelegen
IN AL,DX    ;Wert aus Modemstatusregister
               einlesen
TEST AL,32H ;Test über Bit 5, ob Gegenstelle
               sich meldet
```

Der Assemblerbefehl »TEST« hat den Vorteil, daß er die Registerinhalte nicht verändert.

Allgemein nicht unwichtig sind auch die Register für die Interrupt-Verarbeitung, nämlich das Unterbrechungs-Aktivierungs- und das Unterbrechungs-Erkennungsregister.

Für das DFÜ-Programm »STrans« sind diese beiden Register allerdings nicht weiter von Bedeutung.

(Hartmut Peyke/ma)

Zu guter Letzt muß noch die Anzahl der Stoppbits pro Bitpaket festgelegt werden. Dies geschieht über das Bit 2 des Leitungssteuerregisters. Ist dieses Bit gesetzt, so arbeitet der Baustein 8250 mit zwei, andernfalls mit nur einem Stoppbit.

Auch in diesem Fall muß wieder die Oder-Verknüpfung benutzt werden, um die bereits gesetzten Bits nicht zu überschreiben.

Das folgende Beispiel stellt für die Datenübertragung zwei Stoppbit ein:

```
MOV DX,3FBH ;Leitungssteuerregister vorbelegen
MOV AH,04H  ;Bit 2=1 für zwei Stoppbit
IN AL,DX    ;vorbelegten Wert aus DX in
               AL einlesen
OR AL,AH    ;AL mit AH über Oder-Funktion
               verknüpfen
```

KAT-System

CPU 80286, umschaltbar 6/8/10 MHz, 512 KB Hauptspeicher auf 4 MB aufrüstbar, je 1 ser./par. Schnittstelle, Hercules komp. Karte, 220-W-Netzteil, 1,2 MB Floppy mit Controller, DIN Tastatur, 14" Monitor

DM 2668,-

Aufpreis 20 MB Winchester u. Controller **DM 688,-**

KXT-TURBO-System

CPU 8088, umschaltbar 4,77/8 MHz, 256 KB Hauptspeicher auf 640 KB aufrüstbar, Hercules komp. Grafik Drucker-Karte, 150-W-Netzteil, 360 KB Floppy mit Controller, DIN Tastatur

DM 1054,-

Aufpreis 20 MB Winchester u. Controller **DM 682,-**

Hercules Komp. Karte	DM 109,-	XT I/O Plus Karte	DM 113,-
EGA Karte	DM 339,-	360 KB Floppylaufwerk	DM 217,-
XT Floppycontroller	DM 56,-	1,2 MB Floppylaufwerk	DM 262,-
XT Festplattencontr.	DM 170,-	20 MB Winchester	DM 512,-
AT Festplattencontr.	DM 262,-	DATAS 14" Monitor	DM 268,-
AT Kombicontroller	DM 284,-	EGA Monitor 14"	DM 1015,-
XT Multi I/O Karte	DM 142,-	ADI 14" Monitor	DM 379,-

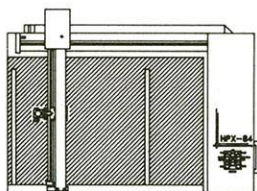
Händleranfragen erwünscht – Autorisierter Schneider-Händler

KLEINELECTRONIC

Biskup u. Broicher

Hermannstraße 18, 4050 Mönchengladbach 1, Telefon: 021 61/2 10 13-14

Plotter zu vernünftigen Preisen



DIN A3 Flachbettplotter HPX-84

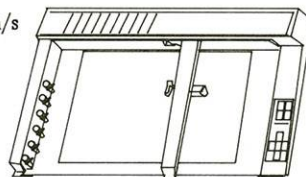
Technische Daten:
 Plottfläche 290 x 390mm
 Wiederholgenauigkeit besser 0.1mm
 Absolute Genauigkeit Abweich. max 1%
 Befehlssatz HP-GL Kompatibel
 Schnittstelle Centronics
 Anzahl der Stifte 1
 Auflösung 0.05mm und 0.025mm

Preis ab DM 1398.-

8-Farben DIN-A3-Flachbettplotter KPL-750

Zeichengeschwindigkeit 400mm/s
 Zeichenfläche 404 x 276mm
 Wiederholgenauigkeit 0.2mm
 Auflösung 0.025mm
 Schnittstellen:
 Centronic und RS-232c
 56 HP-GL Befehle

Preis DM 2348.-



DIN-A1-Flachbettplotter HBS-101



Technische Daten:
 Zeichenfläche 810 x 620mm
 Zeichengeschwindigkeit 120mm/s
 Auflösung 0.025mm
 Wiederholgenauigkeit 0.1mm
 Schnittstelle Centronic
 Option RS-232c
 Befehlssatz HP-GL kompatibel

Preis DM 7498.-

Weitere Information bei

HBS-Grafiksysteme

Leonhard Habersetzer

Rigi Str.35, 8123 Peißenberg, Tel.08803/2670

Oft sind es gar nicht große Programme, die ein Problem lösen helfen, sondern jene kleinen Hinweise, kurze Routinen und Hintergrundinformationen, die in dieser Rubrik von Lesern für Leser vorgestellt werden. Deshalb werden Sie sicher in der DOS Extra immer wieder etwas Nützliches finden. Wenn Sie also einen Tip für andere Programmierer haben oder einen kleinen Trick für das Arbeiten in einer höheren Programmiersprache kennen, dann sollten Sie ihn nicht für sich behalten, sondern allen DOS-Extra-Lesern zugänglich machen. Natürlich gibt es für jeden abgedruckten Beitrag ein angemessenes Honorar. Senden Sie Ihren Tip oder Ihre kurze Routine an:

**Redaktionsbüro Everts & Hagedorn, Redaktion DOS International, Stichwort Tips & Tricks,
Hans-Stießberger-Straße 3, 8013 Haar/München .**

Tips & Tricks

In dieser Rubrik finden Sie alles, was Sie über das Arbeiten mit den verschiedenen Programmiersprachen wissen müssen, um Ihren PC optimal einsetzen zu können. Ob Pascal, C, Basic, Prolog oder Assembler – zu jeder Sprache gibt es viele interessante Hinweise und Tips.

Laufwerksinformationen mit GW-Basic

Manchmal ist es sehr nützlich, nähere Informationen über das Diskettenlaufwerk zu erhalten, sei es, um Fehler zu beseitigen oder um direkt auf die Diskette zu schreiben oder zu lesen. Um dies durchzuführen, sind folgende Befehlsfolgen hilfreich.

Um die aktive Spur zu bekommen, bauen Sie in Ihre Programme folgende Befehle ein:

```
def seg=64:print peek(69)
```

Den zuletzt aktiven Schreib- und Lesekopf erhalten Sie mit:

```
def seg=64:print peek(70)
```

Den Sektor, der zuletzt bearbeitet wurde, erhalten Sie folgendermaßen:

```
def seg=64:print peek(71)
```

Die Anzahl der Bytes pro Sektor können Sie so herausbekommen:

```
def seg=64:print 128*2↑peek(72)
```

Um zu testen, ob ein Schreibschutz vorhanden ist, lautet die Befehlsfolge:

```
def seg=64:if (peek(66) and 3)=3 then print "Ja." else print "Nein."
```

(Arne Bert Steingraber/gh)

Einfacher Datenschutz mit GW-Basic

Basic und MS-DOS benutzen das ASCII-Zeichen 26 als Endezeichen für Dateien. Das ASCII-Zeichen 26 ist Ihnen sicher von den Batch-Dateien her bekannt, denn auch hier schließen Sie die Datei mit Ctrl-Z. Wird dieses Zeichen in eine sequentielle Datei geschrieben, kann alles, was auf dieses Zeichen folgt, nicht mehr gelesen werden. Anders verhält es sich jedoch mit Random-Dateien. Alles was hinter dem Zeichen »chr\$(26) steht, kann zwar mit dem Befehl »type« nicht mehr auf den Bildschirm geholt werden, das Lesen der kompletten Datei durch ein Programm ist jedoch kein Problem. Als einfach und trotzdem effektiv erweist sich folgende erste Zeile in einer Random-Datei:

```
print #1, "Geheim!!!"; chr$(26)
```

Danach werden die Daten wie gewohnt in die Datei geschrieben, man kann jedoch außer der ersten Zeile nichts mehr auf den Bildschirm oder den Drucker ausgeben.

(Arne Bert Steingraber/gh)

Speichern des Bildschirms mit GW-Basic

Den aktuellen Inhalt des Bildschirms kann man auf sehr einfache Art auf die Diskette speichern. Durch den Befehl »bsave« speichern und lesen Sie den Inhalt des Bildschirmspeichers direkt. Der Befehl »bload« kann dann wiederum die Daten zurückholen. Jedoch müssen Sie bei beiden Befehlen die Anfangsadresse des zu bearbeitenden Speicherblocks angeben und bei dem Befehl »bsave« die Endadresse des Speicherblocks (hier 4000) festlegen. Als erstes das Speichern des Bildschirminhalts:

```
10 def seg=&HB800  
20 bsave "schirm.001",0,4000
```

Von der Diskette gelesen wird der Inhalt durch:

```
10 def seg=&HB800  
20 bload "schirm.001",0
```

Wenn Sie eine Herculeskarte besitzen, müssen Sie jeweils in der Zeile 10 den Befehl durch »def seg = &HB000« ersetzen.

(Arne Bert Steingraber/gh)

Die Systemkonfiguration mit GW-Basic ermitteln

Oft ist es erforderlich, die Ausstattung eines Systems durch ein Programm erkenntlich zu machen. Dies hat vor allem dann Vorteile, wenn man sichere Programme schreiben will, die nicht einfach verschwinden, weil beispielsweise ein Peripheriegerät nicht vorhanden ist. Oder man möchte vor dem Start des eigentlichen Programms prüfen, ob der verfügbare freie Speicherplatz ausreicht. Das Bestimmen der Systemkonfiguration ist mit Basic keine unüberwindliche Hürde. Wenn Sie die Anzahl der angeschlossenen Diskettenlaufwerke wissen müssen, helfen folgende Befehle weiter:

```
10 def seg=0  
20 ? "Anzahl Diskdrives: ";  
30 if peek(1040) and 1=0 then ? "Keine.": end  
40 ? ((peek(1040) and 192)/64)+1
```

Zeile 40 von diesem kleinen Programm ermittelt die Anzahl der Laufwerke. Die Anzahl der Druckeradapter läßt sich folgendermaßen feststellen:

```
10 def seg=0  
20 ? "Anzahl Druckeradapter: ";  
30 ? (peek(1041) and 192)/64
```


Die Anzahl der RS232-Adapter ermitteln Sie so:

```
10 def seg=0
20 ? "Anzahl RS232-Adapter: ";
30 ? (peek(1041) and 16)/16
```

Sie können auch die Anzahl der Spieleadapter feststellen. Das Programm funktioniert allerdings auf einem AT nicht:

```
10 def seg=0
20 ? "Anzahl Spieleadapter: ";
30 ? (peek(1041) and 14)/2
```

Den Computertyp finden Sie mit dem nächsten Programm heraus:

```
10 def seg=&HF000
20 typ=peek(&HFFFE)
30 if typ=&HF8 then ? "PS/" else if typ=&HFF then ? "PC" else if
typ=&HFE then ? "XT" else if typ=&HFD then ? "JR" else if typ=&hfc
THEN ? "at" ELSE ? "Unbekannt!"
```

Zum Schluß noch drei Basiczeilen, die den freien Speicherplatz ausgeben. Dieses Programm unterstützt aber keine EMS-Anforderung:

```
10 def seg = 0
20 ? "Freier Speicherplatz: ";
30 ? peek(1043)+peek(1044)*256
```

(Arne Bert Steingraber/gh)

Laufschriften mit Turbo Pascal

Wenn man ein Programm fertiggestellt hat, steht man meist vor dem Problem der Titelbild-Gestaltung. Besonders reizvoll ist hier eine Laufschrift. Mit dem Programm »lauf.pas« (Listing 1) können Sie jeden Text mit dem gewünschten Rahmen je nach Bedarf an bestimmten Bildschirmpositionen ein- oder ausblenden. Mit der Routine »lauf_1« läuft der Text vom rechten Bildschirmrand ab der Position »startpos« bis zur Position »endpos« am linken Rand. Die Routine »lauf_2« macht dasselbe, jedoch von links nach rechts. »lauf_3« läßt den Text wieder von links nach rechts wandern, jedoch verschwindet der Text an der Position »endpos«. »lauf_4« erledigt wieder das gleiche, jedoch von rechts nach links. »stop« bringt den Text blinkend ab der Position »endpos« auf den Bildschirm. Sicher ist auch für Sie eine Routine enthalten, die Sie verwenden können.

(Joachim Hanke/gh)

```
[3U5] 0: PROGRAM laufschrift;
[000] 1:
[ZT5] 2: TYPE lang_80 = STRING[80];
[000] 3:
[XK4] 4: VAR a,b,c : INTEGER;
[R22] 5: text_1 ,
[MT4] 6: text_2 : STRING[80];
[000] 7:
[P1K] 8: PROCEDURE lauf_1(text : LANG_80;posy2,start
pos,endpos : INTEGER);
[B21] 9: BEGIN
[2UE] 10: { * Laufschrift von rechts nach links aufb
auend * }
[Z79] 11: { * startpos > endpos
* }
[1D1] 12: b := 0;
[AGB] 13: FOR a := startpos DOWNT0 endpos DO BEGIN
[V88] 14: IF b < LENGTH(text) THEN b := SUCC(b);
[GV4] 15: gotoxy(a,posy2);
[576] 16: write(COPY(text,1,b));
[TM0] 17: END;
[TM0] 18: END;
[000] 19:
[P1K] 20: PROCEDURE lauf_2(text : LANG_80;posy2,start
pos,endpos : INTEGER);
[B21] 21: BEGIN
[F3F] 22: { * Laufschrift von links nach rechts aufb
auend * }
[T79] 23: { * startpos < endpos
* }
[ZB4] 24: b := LENGTH(text);
[2D1] 25: c := 0;
[3C9] 26: FOR a := startpos TO endpos DO BEGIN
[7M2] 27: b := PRED(b);
[GV4] 28: gotoxy(a,posy2);
[TK7] 29: IF a > (LENGTH(text)-1) THEN BEGIN
[HN2] 30: c := SUCC(c);
[GV4] 31: gotoxy(c,posy2);
[2A3] 32: write(text);
[DD0] 33: END
```

```
[YY1] 34: ELSE BEGIN
[6S7] 35: gotoxy(startpos,posy2);
[L86] 36: write(COPY(text,b,a));
[TM0] 37: END;
[TM0] 38: END;
[TM0] 39: END;
[000] 40:
[P1K] 41: PROCEDURE lauf_3(text : LANG_80;posy2,start
pos,endpos : INTEGER);
[B21] 42: BEGIN
[4GE] 43: { * Laufschrift von links nach rechts abba
uend * }
[Z69] 44: { * startpos < endpos
* }
[ZB4] 45: b := LENGTH(text);
[3C9] 46: FOR a := startpos TO endpos DO BEGIN
[ST9] 47: IF endpos - a > LENGTH(text) THEN BEGIN
[GV4] 48: gotoxy(a,posy2);
[2A3] 49: write(text);
[NL2] 50: clreol;
[DD0] 51: END
[YY1] 52: ELSE BEGIN
[7M2] 53: b := PRED(b);
[GV4] 54: gotoxy(a,posy2);
[576] 55: write(COPY(text,1,b));
[NL2] 56: clreol;
[TM0] 57: END;
[TM0] 58: END;
[TM0] 59: END;
[000] 60:
[P1K] 61: PROCEDURE lauf_4(text : LANG_80;posy2,start
pos,endpos : INTEGER);
[B21] 62: BEGIN
[R6E] 63: { * Laufschrift von rechts nach links abba
uend * }
[Z69] 64: { * startpos < endpos
* }
[ZB4] 65: b := LENGTH(text);
[AGB] 66: FOR a := startpos DOWNT0 endpos DO BEGIN
[GV4] 67: gotoxy(a,posy2);
[2A3] 68: write(text);
[NL2] 69: clreol;
[TM0] 70: END;
[2D1] 71: c := 0;
[CEA] 72: FOR a := LENGTH(text) DOWNT0 endpos DO BE
GIN
[7M2] 73: b := PRED(b);
[HN2] 74: c := SUCC(c);
[286] 75: gotoxy(endpos,posy2);
[R86] 76: write(COPY(text,c,b));
[NL2] 77: clreol;
[TM0] 78: END;
[TM0] 79: END;
[000] 80:
[6UD] 81: PROCEDURE stop(text : LANG_80;posy2,endpos
: INTEGER);
[B21] 82: BEGIN
[286] 83: gotoxy(endpos,posy2);
[VMG] 84: textcolor(15+blink);write(text);delay(200
0);normvideo;
[6U8] 85: gotoxy(1,posy2);write;clreol;
[TM0] 86: END;
[000] 87:
[B21] 88: BEGIN
[82Z] 89: text_1 := '
';
[KYB] 90: text_2 := ' >>>>>> D O S -- Internation
al <<<<<< ';
[000] 91:
[NM2] 92: clrscr;
[000] 93:
[FD8] 94: gotoxy(1,1);writeln(text_1);
[LD8] 95: gotoxy(1,3);writeln(text_1);
[UY4] 96: lauf_1(text_2,2,80,1);
[R54] 97: stop(text_2,2,1);
[000] 98:
[34D] 99: gotoxy((80 - LENGTH(text_1)),5);writeln(t
ext_1);
[F4D] 100: gotoxy((80 - LENGTH(text_1)),7);writeln(t
ext_1);
[D05] 101: lauf_2(text_2,6,1,80);
[F79] 102: stop(text_2,6,(80 - LENGTH(text_2)));
[000] 103:
[SA8] 104: gotoxy(1,9);writeln(text_1);
[4C8] 105: gotoxy(1,11);writeln(text_1);
[B84] 106: stop(text_2,10,1);
[5A5] 107: lauf_3(text_2,10,1,80);
[000] 108:
[UGD] 109: gotoxy((80 - LENGTH(text_1)),13);writeln(
text_1);
[8HD] 110: gotoxy((80 - LENGTH(text_1)),15);writeln(
text_1);
[X69] 111: stop(text_2,14,(80 - LENGTH(text_2)));
[W69] 112: lauf_4(text_2,14,(80 - (LENGTH(text_2))),1)
;
[000] 113:
[5LO] 114: END.
```

Listing 1. Mit diesem Programm können Sie Laufschriften einbauen

Drucker-Unit für Turbo Pascal 4.0

Oft bricht ein Programm, das auf den Drucker zugreift, ab, weil dieser nicht zur Ausgabe bereit ist (beispielsweise wenn das Papier ausgegangen ist, oder der Drucker auf »off-line« geschaltet ist). Die einzige Alternative bestand bisher darin, mit Hilfe der Compileroption »\$I-« die Fehlerüberprüfung für die Ein- und Ausgabe auszuschalten und mit der Funktion »ioresult« jede Ausgabe auf ihre Korrektheit zu überprüfen. Dies ist aber sehr umständlich, besonders dann, wenn man viele Ausgabezeilen an den Drucker im Programm hat. Turbo Pascal 4.0 erlaubt es, eigene Treiber für Textdateien zu installieren (Listing 2). Zunächst wurde der Typ »textrec« geändert, um zusätzliche Informationen wie die Nummer der Schnittstelle (Druckernr) aufzunehmen oder festzustellen, ob der Drucker zu Beginn initialisiert werden soll (init). Den Kern des Programms bilden die Routinen »prn_output«, »prn_close« und »prn_open«. Sie ersetzen die ursprünglichen Funktionen. Die Zuordnung zu einer Textdatei geschieht mit der Prozedur »assign_prn«. Hier werden als Parameter Variable vom Typ »text«, »word« und boolean erwartet. Diese Prozedur setzt die Zeiger für die Ausgabe-funktionen neu.

Die Funktion »prn_output« gibt die in dem Puffer gespeicherten Zeichen an den Drucker weiter, wobei die Routine

die Prozedur »prn_out« aufruft. Diese sendet mit Hilfe des Interrupts \$17 ein Zeichen an den Drucker und liefert den Druckerstatus zurück. Tritt ein Fehler auf, so wird die Funktion »fehler« aufgerufen. Diese Funktion liefert den Wert »true« zurück, wenn Sie den Ausdruck fortsetzen wollen. Andernfalls wird der Wert »false« gesetzt. Die Funktion »fehler« können Sie natürlich jederzeit durch eine eigene Fehlerbehandlungsroutine ersetzen. Diese Routine bekommt eine Variable vom Typ »word« zugewiesen, die den aufgetretenen Fehler angibt, und verwendet dafür definierte Konstanten. Die Funktion muß »true« zurückgeben, wenn man den Ausdruck fortsetzen will. Auch die Prozedur »ausgabe_melden« kann man durch eine eigene ersetzen oder aber ganz weglassen. Sie macht den Benutzer lediglich darauf aufmerksam, daß man im Moment Daten an den Drucker sendet. Die Funktion »prn_out«, die ein Zeichen an den Drucker ausgibt und die Prozedur »prn_init« sind aus Geschwindigkeitsgründen mit Inlinecode geschrieben.

Ist die Ausgabe abgebrochen worden, so können Sie erst wieder Daten an den Drucker senden, wenn Sie die Datei mit einem »rewrite« erneut öffnen. Davor muß die Datei aber geschlossen werden! Das Beispielprogramm (Listing 3) zeigt Ihnen, wie Sie diese Unit einsetzen können.

(Carsten Scherb/gh)

```
[9M3] 0: UNIT drucker;
[Z12] 1: INTERFACE
[1B2] 2: USES crt;
[000] 3:
[SLP] 4: PROCEDURE assign_prn (VAR f : text; drucker
      : word; Initialisierung : boolean);

[000] 5:
[2L3] 6: IMPLEMENTATION
[AS0] 7: TYPE
[VZ8] 8:   TextBuf = ARRAY[0..127] of Char;
[HR4] 9:   TextRec = RECORD
[8S4] 10:      Handle : word;
[NZ3] 11:      Mode : word;
[095] 12:      BufSize : word;
[HB5] 13:      Private : word;
[5T4] 14:      BufPos : word;
[UM4] 15:      BufEnd : word;
[TH5] 16:      BufPtr : ^TextBuf;
[SW5] 17:      OpenFunc : pointer;
[VZ5] 18:      InOutFunc : pointer;
[Z16] 19:      FlushFunc : pointer;
[YZ5] 20:      CloseFunc : pointer;
[CE5] 21:      DruckerNr : word;
[8R4] 22:      Init : boolean;
[TM8] 23:      UserFill : ARRAY[1..13] of b
      yte;
[F97] 24:      Name : ARRAY[0..79] of c
      har;
[XC5] 25:      Buffer : TextBuf;
[TM0] 26:      END;
[000] 27:
[V51] 28: CONST
[AC4] 29:   fmClosed = $D7B0;
[YG4] 30:   fmInput = $D7B1;
[9K4] 31:   fmOutput = $D7B2;
[TB4] 32:   fmInOut = $D7B3;
[000] 33:
[FA4] 34:   fehlerfrei = 0;
[5G4] 35:   kein_papier = 1;
[095] 36:   ausgeschaltet = 2;
[H34] 37:   off_line = 3;
[000] 38:
[000] 39:
[DW7] 40: PROCEDURE ausdruck_melden;
[B21] 41: BEGIN
[U54] 42:   window(19,8,61,16);
[Z7Z] 43:   write('
[MD7] 44:   write('
[MD7] 45:   write('
[MD7] 46:   write('
[MD7] 47:   write('
[MD7] 48:   write('
[MD7] 49:   write('
```

```
[18Z] 50:   write('
[000] 51:   window(20,9,60,15);
[W43] 52:   gotoxy(1,3);
[VC9] 53:   write('      AUSDRUCK GEHT AN DEN DRUCKER
[000] 54:   write('
[R04] 55:   window(1,1,80,25);
[TM0] 56:   END;
[000] 57:
[PNE] 58: FUNCTION fehler (fehlernummer : byte) : boo
      lean;
[EE0] 59: VAR
[UR6] 60:   meldung : string[30];
[PH3] 61:   c,n : char;
[000] 62:
[B21] 63: BEGIN
[626] 64:   CASE fehlernummer of
[WCF] 65:      kein_papier : meldung := 'Kein Papier
      im Drucker';
[94K] 66:      ausgeschaltet : meldung := 'Drucker ist
      ausgeschaltet';
[RME] 67:      off_line : meldung := 'Drucker ist
      off-line';
[TM0] 68:   END;
[37Z] 69:   write(#7);
[U54] 70:   window(19,8,61,16);
[Z7Z] 71:   write('
[MD7] 72:   write('
[MD7] 73:   write('
[MD7] 74:   write('
[MD7] 75:   write('
[V7Z] 76:   write('
[MD7] 77:   write('
[18Z] 78:   write('
[D54] 79:   window(20,9,60,15);
[S43] 80:   gotoxy(1,1);
[UL8] 81:   write('F E H L E R   B E I M   A U S D R
      U C K');
[W43] 82:   gotoxy(1,3);
[Z74] 83:   write('URSACHE : ');
[BT4] 84:   write(meldung);
[Z53] 85:   gotoxy(1,6);
[M5B] 86:   write('      F10 = Abbruch      RETURN = weite
      r');
[XG1] 87:   REPEAT
[8E3] 88:      c := readkey;
[8V2] 89:      IF c = #0 THEN
[LE3] 90:         n := readkey;
[H08] 91:   UNTIL (c = #13) or ((c = #0) and (n = #68
```



```

    ));
[9B4] 92: IF c = #13 THEN BEGIN
[DU5] 93:   ausdruck_melden;
[2W4] 94:   fehler := true;
[DD0] 95:   END
[AR0] 96:   ELSE
[135] 97:   fehler := false;
[TM0] 98: END;
[000] 99:
[000] 100:
[Y3A] 101: PROCEDURE prn_init(drucker : word);
[9T1] 102: INLINE(
[5V7] 103:   $5A/      {Druckernummer -> DX}
[VAC] 104:   $B4/$01/   {Functionscode : Initialiseru
ng};
[105] 105:   $CD/$17); {BIOS - Aufruf}
[000] 106:
[000] 107:
[4EE] 108: FUNCTION prn_out(drucker : word; ch : char)
: byte;
[9T1] 109: INLINE(
[DUB] 110:   $58/      {auszugebendes Zeichen -> ax
(AL)};
[GM7] 111:   $5A/      {Druckernummer -> DX}
[Y4D] 112:   $B4/$00/ {Functionscode : Zeichen aus
geben};
[106] 113:   $CD/$17/ {BIOS - Aufruf}
[OR5] 114:   $88/$E0; {Status -> AL}
[000] 115:
[000] 116:
[K51] 117: {$F+}
[NVD] 118: FUNCTION dummy_output (VAR f : TextRec) : i
nteger;
[B21] 119: BEGIN
[OL3] 120:   f.BufPos := 0;
[T55] 121:   dummy_output := 0;
[TM0] 122: END;
[000] 123:
[000] 124:
[MBD] 125: FUNCTION prn_output (VAR f: TextRec) : inte
ger;
[5R1] 126: LABEL 0 ;
[000] 127:
[V51] 128: CONST
[39A] 129:   wert : ARRAY[0..7] of byte = (1,2,4,8,16,
32,64,128);
[000] 130:
[EE0] 131: VAR
[AYA] 132:   status : ARRAY[0..7] of bo
olean;
[NA8] 133:   fehlernummer,j,error : byte;
[UT5] 134:   p : integer;
[000] 135:
[B21] 136: BEGIN
[DM3] 137:   WITH f DO BEGIN
[4Z3] 138:     IF BufPos > 0 THEN
[YC5] 139:       FOR p := 0 to BufPos-1 DO
[XG1] 140:         REPEAT
[EZE] 141:           fehlernummer := prn_out(DruckerNr
,BufPtr^[p]);
[000] 142:
[DL6] 143:           {Fehlerüberprüfung}
[4H5] 144:           FOR j := 7 downto 0 DO
[RCB] 145:             IF (fehlernummer - wert[j]) >=
0 THEN BEGIN
[XOC] 146:               fehlernummer := fehlernummer
- wert[j];
[075] 147:               status[j] := true
[DD0] 148:             END
[AR0] 149:             ELSE
[YC5] 150:               status[j] := false;
[000] 151:
[8S4] 152:           {Überprüfung}
[PWE] 153:           IF not status[3] THEN {kein F
ehler aufgetreten}
[BB2] 154:             error := 0
[YY1] 155:           ELSE BEGIN
[E46] 156:             {Fehlerbehandlung}
[HP4] 157:             IF status[5] THEN
[HT7] 158:               error := 1 {kein P
apier}
[AR0] 159:           ELSE
[RP4] 160:             IF status[7] THEN
[HAB] 161:               error := 2 {Drucke
r ausgeschaltet}
[AR0] 162:             ELSE
[10A] 163:               error := 3 {Drucke
r off-line}
[TM0] 164:           END;
[000] 165:
[E46] 166:           {Fehlerbehandlung}
[7B4] 167:           IF error <> 0 THEN
[F5G] 168:             IF not fehler(error) THEN BEGI
N
[62K] 169:               InOutFunc := @dummy_output;
{erneute Ausgabe erst nach }
[YVH] 170:               FlushFunc := @dummy_output;
{einem 'rewrite' möglich}
[UX1] 171:               goto 0;

```

```

[7C1] 172:           exit;
[TM0] 173:           END;
[RL3] 174:           UNTIL error = 0;
[MU2] 175:           BufPos := 0;
[TM0] 176:           END;
[BE5] 177:           0 : prn_output := 0;
[TM0] 178:           END;
[000] 179:
[D4D] 180: FUNCTION prn_close (VAR f : TextRec) : inte
ger;
[B21] 181: BEGIN
[Y94] 182:   prn_close := 0;
[TM0] 183:   END;
[000] 184:
[000] 185:
[2XC] 186: FUNCTION prn_open(VAR f : TextRec) : intege
r;
[B21] 187: BEGIN
[DU5] 188:   ausdruck_melden;
[DM3] 189:   WITH f DO BEGIN
[C83] 190:     IF Init THEN
[1A6] 191:       prn_init(DruckerNr);
[LL5] 192:     mode := fmOutput;
[NS7] 193:     InOutFunc := @prn_output;
[WT7] 194:     FlushFunc := @prn_output;
[YG7] 195:     CloseFunc := @prn_close;
[TM0] 196:     END;
[784] 197:     prn_open := 0;
[TM0] 198:   END;
[U51] 199:   {$F-}
[000] 200:
[000] 201:
[5LP] 202: PROCEDURE assign_prn (VAR f : text; drucker
: word; Initialisierung : boolean);
[B21] 203: BEGIN
[AY5] 204:   WITH TextRec(f) DO BEGIN
[TD5] 205:     Mode := fmClosed;
[OT7] 206:     BufSize := SizeOf(Buffer);
[E55] 207:     BufPtr := @Buffer;
[967] 208:     OpenFunc := @prn_open;
[YG7] 209:     CloseFunc := @prn_close;
[7B6] 210:     DruckerNr := drucker;
[AR7] 211:     Init := Initialisierung;
[1H3] 212:     Name[0] := #0;
[TM0] 213:   END;
[TM0] 214: END;
[5L0] 215: END.

```

Listing 2. Diese Drucker-Unit verwaltet auch mehrere Drucker

```

[CV4] 0: PROGRAM ausgabe;
[3ZD] 1: USES drucker; { Aufnahme des Units druck
er}
[000] 2:
[EE0] 3: VAR
[PG3] 4:   datei : text;
[000] 5:
[B21] 6: BEGIN
[P9N] 7:   assign_prn(datei,0,true); {Zuweisung der
Variablen datei an den
Drucker und Lpt
1 mit Initialisierung vor
dem Ausdruck}
[Z54] 9:
[2XG] 10: rewrite(datei); {öffnen der Dat
ei zum Schreiben}
[000] 11:
[876] 12: {Ausgabe des Textes}
[14N] 13: writeln(datei,'Dies Ausgabe wird an den D
rucker an Lpt1 geleitet');
[1Z9] 14: writeln(datei,'Dies ist Zeile 2');
[F3B] 15: writeln(datei,'Dies die nächste. ');
[AZ8] 16: write(datei,'Dies Worte und ');
[RFF] 17: writeln(datei,'diese erscheinen in einer
Zeile');
[XH3] 18: close(datei);
[5L0] 19: END.

```

Listing 3. Dieses Programm zeigt Ihnen, wie man die Unit einbaut

Kopieren ohne »copy« mit Turbo Pascal

Oft wird die MS-DOS-Funktion 56 (zum Verschieben von Dateien) in vielen Programmen sträflich vernachlässigt. Eigentlich stellt dies gar kein Problem dar. Genau den gleichen Effekt hat nämlich die Funktion »rename« von Turbo Pascal. Um ein Programm zu verschieben, genügt ein kurzes Programm wie in Listing 4. Als Prozedur aufgerufen kann man es leicht in Programme einbauen.

(Norbert Dohmen/gh)


```
[000] 0:
[T43] 1: var f : file;
[R83] 2: i : integer;
[BH1] 3: begin
[881] 4: (*$I-*)
[FZ4] 5: if paramcount<2
[E7F] 6: then writeln('Aufruf: MOVE <alter-Name> <
neuer-Name>')
[A11] 7: else
[BH1] 8: begin
[AD6] 9: assign(f,paramstr(1));
[VA6] 10: rename(f,paramstr(2));
[7R2] 11: i:=IORESULT;
[1P1] 12: if(i=0)
[ATM] 13: then writeln('Datei von ',paramstr(1),'
nach ',paramstr(2),' umgehängt.')
[L7D] 14: else writeln('Fehler bei Move; Fehler-C
ode: ',i);
[TU0] 15: end;
[5T0] 16: end.
```

Listing 4. Dieses kleine Turbo-Pascal-Programm verschiebt Dateien

Interrupt-Prozeduren mit Turbo C

Um mit dieser Prozedur zu arbeiten, benötigen Sie die Include-Datei »dos.h« für die Aufrufe »setvect« und »getvect«. Die Datei »conio.h« ist nur wegen des »getch()« an Stelle des Hauptprogramms angegeben. Die Variable »akttime« zeigt auf die Adresse 40:6Chex, der Uhrzeit im DOS-Bereich. »gr_mode« auf 40:49hex, also dem aktuellen Video-Modus. Die Variable »screenptr« wird bei Hercules-karten (entspricht dem Video-Modus 7) auf B000:0000hex gesetzt, sonst auf B800:0000hex (für CGA, EGA, VGA). Das Hauptprogramm liest den alten Interrupt-Vektor 1C. Dieser Zeit-Interrupt wird 18,2 Mal je Sekunde aufgerufen und sichert den Eintrag in der Variablen »oldint«. Dann wird die Routine »time_display« als Interrupt 1Chex eingehängt. Ab hier wartet das Programm auf eine beliebige Eingabe von der Tastatur (hier kann dann stattdessen Ihr Hauptprogramm stehen) und setzt danach den Interrupt-Vektor zurück. Es lassen sich natürlich auch speicherresidente Programme schreiben, also Interrupt-Handler, die bei Programmende im Speicher bleiben. Dazu wird die Funktion »keep« aufgerufen (siehe Seite 141 im Reference Guide). Die Funktion erwartet als ersten Parameter einen Fehlerstatus (normalerweise 0), als zweiten Parameter die Größe des Speicherbereiches, der resident bleiben soll. Dieser Parameter ist nur mit genauer Kenntnis des verwendeten Memory-Modells zu ermitteln (beschrieben im Users Guide von Turbo C ab Seite 234). Der einfachste Weg ist, das Programm mit eingeschalteter Map-Option (Publics on) zu binden. Der Ladeliste kann man dann entnehmen, welche Speicherbereiche vom residenten Teil benötigt werden.

Durch den Zusatz »interrupt« erzeugt der Compiler zu Beginn der Interrupt-Routine Push-Befehle zur Sicherung der Register. Am Ende der Funktion werden die Register mit »pop« zurückgeholt und die Funktion statt mit einem »ret« mit einem »iret«, daß auch noch die Flags vom Stack holt, beendet.

Der Routine werden alle Register als Parameter übergeben. Da die Routine aber vom Type »__cdecl« sein muß, brauchen Sie nur die Register anzugeben, deren Werte man lesen oder verändern will. Listing 6 benötigt keine Register.

Da MS-DOS nicht reentrant geschrieben ist, darf die Interrupt-Routine selbst keine Interrupts erzeugen, seien Sie also vorsichtig mit Funktionsaufrufen innerhalb Ihres Interrupt-Codings! Der Interrupt von Listing 6 liest den Zeiteintrag aus dem MS-DOS-Bereich, wandelt diesen Zeit-

eintrag durch die Division mit 18,2 in einen Sekundenwert um, und legt ihn dann in der lokalen Variablen »newtime« ab. Nur wenn dieser Wert in der Variablen »newtime« vom Wert des vorigen Durchlaufs abläuft, werden hieraus die Stunden, Minuten und Sekunden extrahiert und auf dem Bildschirm ausgegeben.

Für die Bildschirmausgabe ist natürlich der Aufruf »printf« verboten, weil der Befehl den Interrupt 21h aufruft. Die Ausgabe kann also nur durch die direkte Manipulation des Bildschirmpuffers erfolgen. Bei Computern mit einer Farbgrafikkarte kann dadurch Schnee auf dem Bildschirm erscheinen; da aber nur ein Schreibvorgang je Sekunde erfolgt, führt dies nicht zu starken Störungen. Wer aber damit nicht zufrieden ist, kann die Routine »mcmvsmem« verwenden, die zum Lieferumfang von Turbo C gehört und sich im Programm »Mcalc« befindet. Oder Sie warten vor jedem Schreiben das Synchronisations-Signal ab, lesen also den Port 3DA, bis das Low-Bit gesetzt ist, lesen dann wieder den Port, bis das Low-Bit gelöscht ist.

(Norbert Dohmen/gh)

```
[C85] 0: #include <dos.h>
[7V5] 1: #include <conio.h>
[000] 2:
[R4C] 3: long oldtime=0; /* vorherige
Zeit */
[V7K] 4: long far *akttime=(long far *)0X0040006CL;
/* => DOS-Uhrzeit (40:6C) */
[12H] 5: char far *gr_mode=(char far *)0X00400049L;
/* => Grafik-Modus */
[DFE] 6: char far *screenptr; /* Bildsch
irm-Zeiger */
[23K] 7: void interrupt far (*oldint)(); /*
alter timer-interrupt */
[000] 8:
[TOL] 9: void interrupt time_display() /* Ti
mer-Interrupt-Routine */
[V30] 10: {
[RUA] 11: long newtime; /* aktuelle Z
eit */
[000] 12:
[F4A] 13: if((newtime=(*akttime/18.2))!=oldtime)
[V30] 14: {
[CV5] 15: oldtime=newtime;
[CH8] 16: *(screenptr+158)='0'+(newtime%10);
[VL6] 17: newtime=(newtime/10);
[1B8] 18: *(screenptr+156)='0'+(newtime%6);
[HC6] 19: newtime=(newtime/6);
[2G8] 20: *(screenptr+152)='0'+(newtime%10);
[VL6] 21: newtime=(newtime/10);
[R98] 22: *(screenptr+150)='0'+(newtime%6);
[HC6] 23: newtime=(newtime/6);
[SG8] 24: *(screenptr+146)='0'+(newtime%10);
[VL6] 25: newtime=(newtime/10);
[DA8] 26: *(screenptr+144)='0'+(newtime%6);
[X30] 27: }
[X30] 28: }
[000] 29:
[0F1] 30: main()
[V30] 31: {
[2SF] 32: screenptr=(char far *)(*gr_mode==7 ? 0XB0
000000L : 0XB8000000L);
[SHA] 33: /* Adresse des Bildschirmpuf
fers */
[3WF] 34: oldint=getvect(0X1C); /* alten In
terrupt sichern */
[D6K] 35: setvect(0X1C,time_display); /* time_
display einhängen */
[SHD] 36: getch(); /* hier steht das übr
ige Programm */
[0UG] 37: setvect(0X1C,oldint); /* Original
-Interrupt setzen */
[X30] 38: }
[000] 39:
```

Listing 5. Dieses Programm gibt die Systemzeit aus

Kein Parkverbot

Ein Headcrash ist wohl der Alptraum eines jeden Festplattenbesitzers. Doch schon eine kleine Routine bringt Ihnen mehr Sicherheit im Umgang mit der Harddisk.

Obwohl technisch inzwischen sehr weit fortgeschritten, ist die Festplatte auch heute noch eine der empfindlichsten Komponenten des PC. Harte Stöße, wie sie bei einem Transport immer wieder vorkommen, nimmt sie so manches Mal sehr übel. Befindet sich der Kopf zum Zeitpunkt des Stoßes über einer Spur, die mit Daten oder Programmen belegt ist, so kann es passieren, daß die sehr empfindliche Magnetschicht der Platte beschädigt wird. Ein unwiederbringlicher Verlust der auf dieser Spur befindlichen Daten wäre die Folge. Ist es die Spur, auf der sich das Directory befindet, so kann unter Umständen sogar der gesamte Platteninhalt verloren sein. Wer nun kein Backup von seiner Platte gemacht hat, wie man es bei wichtigen Daten sowieso in kurzen Abständen tun sollte, der hat viel Arbeit vor sich, um seine Daten zu rekonstruieren – falls das überhaupt möglich ist. Selbstverständlich haben auch die Hersteller von Festplatten und Controllern dieses Problem erkannt und Abhilfe geschaffen. Dies geschieht in Form der sogenannten Parkspur, auf die der Kopf vor dem Abschalten der Anlage gefahren wird.

Diese Spur wird nicht mit Daten belegt, somit besteht auch keine Gefahr, daß etwas zerstört wird, falls der Kopf die Plattenoberfläche berührt. Üblicherweise ist zu diesem Zweck die innerste Spur reserviert. Je nachdem, welche Kapazität die Platte nun hat und über wieviele Schreib-/Leseköpfe sie verfügt, unterscheiden sich die Nummern der Parkspuren. Da die kompatiblen Taiwan-PCs oft leider nur mangelhaft dokumentiert sind, finden Sie in der Tabelle eine Auflistung der gängigsten Festplatten mit Angaben über deren Speicherkapazität, die Anzahl der Schreib-/Leseköpfe sowie die Nummer der Parkspur. Wenn Sie Ihre Platte in dieser Liste nicht finden, so sollten Sie sich mit Ihrem Händler in Verbindung setzen, da dieser in der Regel über die entsprechenden technischen Unterlagen verfügt.

Die Betriebssysteme der handelsüblichen Controller sind alle nach einem ähnlichen Muster gestrickt. Die Routine zum Vorformatieren der Platte befindet sich bei C800:5, die zum Anfahren der Parkspur bei C800:4B0. Um nun den Kopf zu parken, muß man eine recht umständliche Proze-

dur durchführen. Auf Betriebssystem-Ebene wird das Programm Debug aufgerufen, von dort aus die entsprechende Unteroutine angewählt und dann die Daten zum Parken des Kopfes eingegeben.

```
C> debug
-g=c800:4b0
Which drive to be locked (0/1) ? 0
Enter the desired cylinder to be locked
(in decimal) 612
Drive locked in the specified cylinder.
C>
```

Ein Tippfehler bei der Eingabe der Sprungadresse kann hier bereits zum Absturz des Computers führen. MS-DOS stellt dem Benutzer ein sehr interessantes, aber leider nur selten verwendetes Hilfsmittel zur Verfügung, das sogenannte »piping«. Dieser Begriff bezeichnet die Option, Ein- und Ausgabeoperationen nicht nur über Tastatur oder Bildschirm durchzuführen, sondern auch andere Geräte zu verwenden. Selbst Textdateien lassen sich so zur Eingabe einsetzen. Das bedeutet, daß zur Durchführung der beschriebenen Aufgabe lediglich zwei Dateien notwendig sind. Der Zweckmäßigkeit halber nennen wir diese Dateien im nachfolgenden Text »parken.bat« und »parken.dat«.

Bei »parken.bat« handelt es sich um eine Batch-Datei, die vom Benutzer aufgerufen wird, um den Kopf zu parken. »parken.dat« beinhaltet die Daten, die »parken.bat« benötigt, um seine Aufgabe ordnungsgemäß durchzuführen.

Beide Dateien sind nur sehr kurz und können daher ohne weiteres mit dem zeilenorientierten Editor Edlin eingegeben werden (Bild 1 und 2).

Die Bedeutung der einzelnen Zeilen der Dateien ist leicht zu durchschauen: Im »Programm« »parken.bat« wird der Bildschirm gelöscht, danach eine Meldung ausgegeben und schließlich Debug aufgerufen. Dabei wird nicht die Tastatur als Eingabemedium aufgerufen, sondern die Textdatei »parken.dat«.

»parken.dat« ist noch kürzer. In der ersten Zeile befindet sich die Sprungadresse, bei der das Unterprogramm zum Parken des Kopfes beginnt. Die zweite Zeile beinhaltet die Nummer der Parkspur. Je nach Festplattenmodell kann diese Nummer differieren. Richten Sie sich hier bitte nach Ihrem Handbuch oder der Tabelle. Die Eingabe der Laufwerksnummer (0/1) läßt

sich bei den meisten Controllern nur manuell vornehmen, deshalb wurde sie in dieser Datei weggelassen. Bei Bedarf können Sie sie jedoch problemlos einfügen.

Nach Ablauf der Batch-Datei vermeiden Sie bis nach dem Transport bitte einen erneuten Zugriff auf die Platte, etwa durch Aufrufen des Directories oder anderer Batch-Dateien, da sonst der Kopf wieder seine Ruheposition verläßt und auf Spuren zugreift, die Daten beinhalten.

(Markus Kersten/ue)

```
echo off
cls
echo Achtung: Der Kopf wird auf die
echo Parkspur gefahren.
echo Nach Abschluß der Operation
echo bitte keine weiteren Arbeiten
echo durchführen, sondern
echo die Anlage direkt ausschalten.
echo -----
echo
echo
debug < parken.dat
```

Bild 1. Die Datei »parken.bat«

```
g=c800:4b0
612
```

Bild 2. Die Datei »parken.dat«

Modell	Typ	Kapazität	Köpfe	Parkspur
EPSON	HD-850	10 MB	4	306
EPSON	HD-860	20 MB	4	612
FUJITSU	M-2233	10 MB	4	306
NEC	D-5114	5 MB	2	306
NEC	D-5124	10 MB	4	306
NEC	D-5126	20 MB	4	612
NEC	D-5146	40 MB	8	612
SEAGATE	ST-212	10 MB	4	306
SEAGATE	ST-213	10 MB	2	612
SEAGATE	ST-225	20 MB	4	612
SEAGATE	ST-4026	20 MB	4	612
SEAGATE	ST-4038	32 MB	5	733
SEAGATE	ST-4051	42 MB	5	977
SHUGART	SA-706	5 MB	2	306
SHUGART	SA-717	10 MB	4	306
TANDON	TM-502	10 MB	4	306
TANDON	TM-261	10 MB	2	612
TANDON	TM-361	10 MB	2	612
TANDON	TM-262	20 MB	4	612
TANDON	TM-362	20 MB	4	612
TANDON	TM-702AT	20 MB	4	612
TANDON	TM-703	30 MB	5	695
TANDON	TM-703AT	32 MB	5	733
TEAC	SD-510	10 MB	4	306
TEAC	SD-520	20 MB	4	612

Tabelle wichtiger Festplattendaten

Die Hercules-Grafik-Toolbox

Haben Sie einen PC mit Hercules-Grafikkarte? Dann suchen Sie bestimmt eine Toolbox, die unter Turbo Pascal Grafiken auf dieser Karte zeichnet.

Ist Ihr PC mit einer »echten« Hercules-Karte ausgerüstet, so kommen Sie mit diesem Programm zum Abtippen in den Genuß eines Grafik-Pakets, das die Leistungsfähigkeit so mancher professionellen Grafik-Programme in den Schatten stellt. Leider läuft das Programm nicht auf Computern, die die Hercules-Grafik nur emulieren, da »hercules.com« direkt auf die Register der Hercules-Karte zugreift. Zu diesen Personalcomputern zählt auch der Schneider PC 1640, so daß das Programm auf diesem Gerät nicht läuft.

Im Lieferumfang von »Turbo Pascal« sind die zwei Programme »graph.p« und »graph.bin« zur Unterstützung der Bildschirmgrafik enthalten. Diese arbeiten jedoch nur mit einer Farbgrafik-Karte. Zur Grafikausgabe auf der Hercules-Karte bieten diverse Hersteller spezielle Programme an. Viele dieser Programme haben aber, abgesehen von ihrem stolzen Preis, schwerwiegende Nachteile. Einige vernachlässigen die Textausgabe im Grafikmodus, andere verzichten auf Cursor und Scrollfunktion, während wieder andere die Fenstergrenzen nicht beachten. Es sind auch Programme darunter, die keine Funktion zum Füllen umrandeter Flächen besitzen, oder aber es können zwar helle und dunkle Linien gezeichnet werden, wobei jedoch der bisherige Bildinhalt an diesen Stellen verlorengeht. Hier ist nämlich eine zusätzliche Funktion notwendig, die beim Linienzeichnen den Bildschirminhalt invertiert. Ei-

nige Grafik-Pakete beanspruchen gar 20 bis 30 KByte des verfügbaren Speicherplatzes (von den 64 KByte für Turbo Pascal-Quellprogramme). Meist verlängert sich auch die Zeit beim Compilieren erheblich. Alle diese Nachteile resultieren aus der fast ausschließlichen Programmierung in Pascal.

Das hier vorgestellte Treiberprogramm für die Hercules-Karte ist in Assembler geschrieben und deshalb äußerst schnell – und vergleichsweise kurz. Dennoch besitzt der Treiber alle wesentlichen Funktionen. Die oben erwähnten Mängel der bestehenden Programme sind hingegen nicht vorhanden.

Das Programmpaket besteht aus drei Teilen:

- Die Grafik-Unterprogramme sind im Speicherresidenten Teil »hercules.com« (Listing 3) zusammengefaßt. Der zugehörige Quellcode wurde mit Rücksicht auf den Editor in die beiden Teile »hercul1.asm« (Listing 1) und »hercul2.asm« (Listing 2) aufgeteilt. Für die Grafikausgabe mit Assembler-Programmen genügt die Datei »hercules.com«.
- »hires.bin« (Listing 5 – der Quellcode steht in Listing 4)) wandelt die von »Turbo Pascal« angelieferten Daten in das von »hercules.com« benötigte Format um.
- »hires.pas« (Listing 6) enthält die Prozedur-Definitionen für Turbo Pascal. Dieser Teil muß mit-

tels einer Include-Anweisung in das Anwendungs-Programm eingebunden werden.

Die Prozedur-Namen stimmen soweit wie möglich mit denen der entsprechenden Prozeduren der »Graphics-Toolbox« von Borland und der Include-Datei »graph.p« überein. Bevor Sie diese Funktionen nutzen können, müssen Sie den Befehl

```
{ $I HIRES }
```

in Ihr Programm einfügen. Wenn alphanumerische Zeichen mit einem ASCII-Wert größer 127 (zum Beispiel Umlaute) ausgegeben werden sollen, müssen Sie vor Ihrem Anwendungsprogramm das Utility-Programm »gratfabl« von Ihrer MS-DOS Systemdiskette aufrufen. »demo.pas« (Listing 7) enthält einige Beispiele für die Anwendung der Grafikfunktionen in einem Turbo Pascal-Programm.

Folgende Grafik-Befehle stehen Ihnen mit »hercules.com« zur Verfügung:

InitGraphic;

Dieser Befehl aktiviert den Grafikmodus, wählt die erste Bildschirmseite, löscht den Bildspeicher, setzt die Zeichenfarbe auf weiß, schaltet den Cursor ein und positioniert ihn in die obere linke Ecke des Bildschirmfensters. Das Fenster wird auf maximale Größe gesetzt.

LeaveGraphic;

Dieser Befehl schaltet wieder zurück in den Textmodus und löscht den Bildschirm.

Hires;

»Hires« aktiviert den Grafikmodus.

TextMode;

Der Textmodus wird mit diesem Befehl eingeschaltet.

HiResColor(Farbe:Integer);

Diese Anweisung setzt die Zeichenfarbe. Folgende Parameter sind erlaubt: »White«, »Black« und »Invers«.

SetLineStyle(Bitmuster:Integer);

Mit diesem Befehl können Sie einen beliebigen Linientyp wählen. Folgende Beispiele verdeutlichen Ihnen die Wirkungsweise dieser Anweisung:

```
Bitmuster = 1 – schwach
punktierte Linie
Bitmuster = 4369 – dicht
punktierte Linie
Bitmuster = 7967 – gestrichelte
Linie
Bitmuster = -1 – Linie
wird durchgezogen
```

SetCursor(StartZeile,EndZeile:Integer);

Zur Einstellung des Cursortyps können Sie die Anweisung »SetCursor« verwenden. Der Parameter »StartZeile« enthält die oberste Zeile des Cursor-Felds, der Parameter »EndZeile« die unterste. Auch hierzu einige Beispiele:

```
StartZeile = 7, EndZeile = 8 – Strich-Cursor
StartZeile = 1, EndZeile = 8 – Voll-Cursor
StartZeile = 0, EndZeile = 0 – Cursor abschalten
```

ClearScreen;

Diese Anweisung löscht den Bildschirm.

DisplayScreen(Seite:Integer);

»DisplayScreen« wählt die aktuelle Bildschirmseite für die Anzeige (nur im Grafikmodus). Erlaubt sind nur die Parameter »1« und »2«.

SelectScreen(Seite:Integer);

»SelectScreen« wählt die aktuelle Bildschirmseite für die Zeichenoperationen. Erlaubt sind nur die Parameter »1« und »2«.

CopyScreen;

»CopyScreen« kopiert den Inhalt der durch »SelectScreen« angewählten Seite in die nicht angewählte Seite.

SwapScreen;

Die Anweisung vertauscht den Inhalt der beiden Bildschirmseiten.

Plot(x,y:Integer);

»Plot« setzt an der Stelle (x,y) einen Punkt.

Draw(x1,y1,x2,y2:Integer);
»Draw« zieht eine Linie vom Punkt (x1,y1) zum Punkt (x2,y2).

Circle(x,y,Radius:Integer);
»Circle« zieht einen Kreis mit dem Halbmesser »Radius« um den Punkt (x,y).

DrawEllipse(x,y,xHalbAchse,yHalbAchse:Integer);
»DrawEllipse« dient zum Zeichnen einer Ellipse der Breite 2x »xHalbAchse« und der Höhe 2y »yHalbAchse« mit dem Mittelpunkt (x,y).

FillShape(x,y:Integer);
Eine beliebig umrandete Fläche, in deren Inneren der Punkt (x,y) liegt, wird durch die Funktion »FillShape« ausgefüllt.

Pattern(Var P);
»Pattern« dient zur Definition eines Füllmusters. Der Parameter ist ein Feld mit einer Größe von acht Byte. Diese legen die acht Zeilen einer 8x8-Punkte-Matrix fest, die zum Füllen von Rechtecken mittels »FillPattern« verwendet wird. Das erste Byte entspricht der obersten Zeile der Matrix, die höchstsignifikanten Bits der ersten Spalte.

FillPattern(x1,y1,x2,y2);
Ein Rechteck mit den Eck-Koordinaten (x1,y1) und (x2,y2) wird mit dem zuvor durch »Pattern« festgelegten Füllmuster ausgefüllt. Ein Beispiel soll die beiden Befehle »Pattern« und »FillPattern« verdeutlichen:

```
Array[1..8] Of Byte =
(129,66,36,24,24,36,129);
Begin Pattern(Muster);
FillPattern(10,5,150,90)
End;
```

Das Rechteck zwischen den Zeilen 5 und 90 und den Spalten 10 und 150 wird mit einem Rauten-Muster gefüllt.

DrawString(x,y:Integer;S:MaxString);
»DrawString« gibt eine Zeichenkette ab der Stelle (x,y) aus. Die Stringlänge darf dabei maximal 90 Zeichen betragen. »x« und »y« sind

die Punktkoordinaten der linken oberen Ecke des Zeichenfeldes des ersten Zeichens.

DrawText(x,y,xz,yz:Integer;S:MaxString);
Dieser Befehl arbeitet wie »DrawString«, jedoch in x- und y-Richtung um den Faktor »xz« und »yz« gedehnt. »xz« sollte nicht größer als 8 sein.

ScrollUp(Zeilen:Integer);
»ScrollUp« bewegt den Inhalt des aktuellen Fensters nach oben. Der Parameter »Zeilen« gibt die Größe der Verschiebung in Punkt-Zeilen an.

GraphWindow(x1,y1,x2,y2:Integer);
»GraphWindow« legt die Begrenzung des aktuellen Fensters fest. (x1,y1) sind die Koordinaten der linken oberen, (x2,y2) die der rechten unteren Ecke.

FillWindow(Bitmuster:Integer);
Durch »FillWindow« wird das Fenster mit einem beliebigen Bitmuster ausgefüllt. Auch hierfür einige Beispiele:

Bitmuster = 0 – Fenster löschen
Bitmuster = -1 – Fenster weiß ausfüllen
Bitmuster = 255 – grobe, vertikale Balken
Bitmuster = 21845 – feine, vertikale Balken

InvertWindow;
Diese Funktion invertiert den Inhalt des Fensters.

CopyWindow(Seite,x,y:Integer);
Der Inhalt des aktuellen Fensters wird an eine beliebige Stelle in die Bildschirmseite »Seite« kopiert. »x« und »y« sind die Koordinaten der linken oberen Ecke der Zielfläche.

Graphmode:Boolean;
Diese Funktion überprüft, ob der Grafik- oder der Textmodus aktiviert ist.

GetDotColor(x,y:Integer):Integer;
Diese Funktion ermittelt die Farbe des Punktes an der Stelle »x,y«.

Die Funktionen »Write«, »Writeln«, »Read«, »Readln«, »WhereX«, »WhereY« und »GotoXY« funktionieren nun auch im Grafikmodus.

Die Grafik-Routinen können auch direkt von Assembler-Programmen aufgerufen werden

Die Grafikfunktionen können von Assembler-Programmen aus über den Interrupt »60H« wie BIOS-Routinen aufgerufen werden. Das Register »AH« muß den Funktions-Code enthalten. Der Inhalt der restlichen Register (außer »AX«) bleiben erhalten:

AH = 00hex
Modus setzen:
AL = 0 –> Textmodus
AL > 3 –> Grafikmodus
Der Bildschirm wird nicht gelöscht. Es wird die erste Seite angezeigt.

AH = 01hex
Momentanen Anzeige-Modus ermitteln. Rückgabe des Modus in AL:
AL = 0 –> Textmodus
AL = FFhex –> Grafikmodus

AH = 02hex
Cursortyp einstellen (vergleiche »SetCursor«). Parameter:
CX = oberste Zeile des Cursor-Feldes
DX = unterste Zeile des Cursor-Feldes

AH = 03hex
Die Position des Cursors wird gesetzt (vergleiche »GotoXY«)
CL = x-Koordinate des Cursors
DL = y-Koordinate des Cursors

AH = 04hex
Momentane Cursor-Position ermitteln. Rückgabe der Koordinaten in »AX«:
AL = x-Koordinate
AH = y-Koordinate

AH = 05hex
Zeichenmodus setzen (vergleiche »HiresColor«):
AL = 0 –> Punkte setzen
AL = 1 –> Punkte löschen
AL = 2 –> Punkte invertieren

Profi-Tools für QuickBASIC und Turbo BASIC

Schreiben Sie schnellere, leistungsfähigere und professionellere Programme! Wir helfen Ihnen dabei mit nützlichen Tools.

Zum Beispiel:

- **Toolbox** (Fenster-technik, Menüs, DOS-Funktionen etc.)
- **Relationale Datenbank** mit komfortablem Masken-Editor
- **Grafik-Paket** (Geschäftsgrafik und Zeichensatz-Generator)
- **Maus-Unterstützung** für Ihre Programme

Alle Pakete mit ausführlich dokumentierten Quelltexten und Programmbeispielen. Wo erforderlich, kommen schnelle Assembler-Routinen zum Einsatz. Wollen Sie mehr aus Ihrem BASIC-Compiler heraus holen? 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

**Copy II Option Board
DELUXE DM 299,-**

**PC Tools Deluxe V 4.21
DM 129,-**

**Copy II/PC V4.01
DM 79,-**

**Pitstop II/Macadam
Bumper je DM 49,95**

**Utah Cobol V4.0/
Fortran je DM 99,-**

**Turbo Pascal 4.0/
T-C V1.5 je 169,-**

**Microsoft C-Compiler
V5.0 DM 699,-**

**Aztec C Professional
DM 299,-**

Kostenlose Prospekte auch für Atari ST und Amiga von

CWTG

Computerversand CWTG
Joachim Tiede Bergstr. 13
D-7109 Roigheim
Tel./BTX 0 62 98/30 98
von 17 - 19 Uhr
SCHRIFTLICHE
HÄNDLERANFRAGEN
ERWÜNSCHT

AT-Kompatibel: 1898,-

- 80286-6/8/10/12 MHz (opt. 1MB, 640/384 möglich)
- 512 KB Ram
- 1.2 MB TEAC-Drive
- Gr. Tastatur DOS 3.2 + 3.3 komp.
- Herc. komp. Grafik
- Printerport

obige Ausstattung plus
2898,-
 - 30 MB Harddisk (Seagate)
 - 14" Flatscreen-Monitor (sw o. amber)
 - serielle Schnittstelle

30 MB
 Seagate WD
 Festplattenkit
598,-

40 MB
 ST 251, 6 Köpfe
698,-

Above-Boards 2 MB:
 Hauptspeicher auf 640 KB aufrüstbar -
 EMS kompatibel - in 256 KB Schritten
 bestückbar - 0 KB Ram

XT: 248,- AT: 298,-

EGA-Wonder (800x560) 448,-
3,5" 720 KB für PC's 298,-

PC 1512/1640
 Festplattenkit
 30 MB
 inkl. Einbausatz
678,-

Frank Strauß Elektronik
 St. Marienplatz 7
 6750 Kaiserslautern
 Tel. 06 31 / 1 62 58 u. 1 62 96

Wer 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. (056 51) 87 02
Telex 993 210 dmv d
Telefax 05 65 12 07 18

Grafik

AH = 06hex

Bildschirmseite für Anzeige wählen (vergleiche »DisplayScreen«):
 AL = 1 -> 1. Bildschirmseite
 AL = 2 -> 2. Bildschirmseite

AH = 07hex

Bildschirmseite für Zeichoperationen wählen (vergleiche »SelectScreen«):
 AL = 1 -> 1. Bildschirmseite
 AL = 2 -> 2. Bildschirmseite

AH = 08hex

Aktuelle Bildschirmseite löschen (vergleiche »ClearScreen«).

AH = 09hex

Kopiert den Inhalt der für Zeichenoperationen angewählten Seite in die nicht angewählte Seite (vergleiche »CopyScreen«)

AH = 0Ahex

Inhalt der beiden Bildschirmseiten vertauschen (vergleiche »SwapScreen«).

AH = 0Bhex

Zeichen in »AL« an die momentane Cursor-Position schreiben und Cursor weiterbewegen.

AH = 0Chex

Zeichenkette auf dem Bildschirm ausgeben (vergleiche »DrawString«).
 CX = x-Koordinate
 DX = y-Koordinate
 DS:SI = Zeiger auf Anfang der Zeichenkette. Das 1. Byte der Zeichenkette muß ihre Länge angeben.

AH = 0Dhex

Zeichenkette vergrößert auf dem Bildschirm ausgeben (vergleiche »DrawText«).
 CX = x-Koordinate
 DX = y-Koordinate
 DI = Vergrößerung in x- und y-Richtung (Low-Byte = x-Vergrößerung, High-Byte = y-Vergrößerung)
 DS:SI = Zeiger auf Anfang der Zeichenkette. Das erste Byte der Zeichenkette muß die Länge enthalten.

AH = 0Ehex

Aktuelles Fenster nach oben schieben (vergleiche »ScrollUp«).
 AL = Zahl der Abtastzeilen

AH = 0Fhex

Punkt setzen (vergleiche »Plot«).
 CX = x-Koordinate
 DX = y-Koordinate

AH = 10hex

Farbe eines Punktes ermitteln. Rückgabe der Farbe in »AL« (vergleiche »GetDotColor«).
 CX = x-Koordinate
 DX = y-Koordinate.

AH = 11hex

Linientyp setzen (vgl »SetLineStyle«).
 CX = Bitmuster

AH = 12hex

Linie von (x1,y1) nach (x2,y2) ziehen (vergleiche »Draw«).
 CX = x1-Koordinate
 DX = y1-Koordinate
 DI = x2-Koordinate
 SI = y2-Koordinate

AH = 13hex

Kreis zeichnen (vergleiche »Circle«).
 CX = x-Koordinate des Mittelpunkts
 DX = y-Koordinate des Mittelpunkts
 DI = Radius

AH = 14hex

Ellipse zeichnen (vergleiche »DrawEllipse«).
 CX = x-Koordinate des Mittelpunkts
 DX = y-Koordinate des Mittelpunkts
 DI = x-Halbachse
 SI = y-Halbachse

AH = 15hex

Beliebige, umrandete Figur ausfüllen (vergleiche »FillShape«).
 CX = x-Koordinate des Punktes innerhalb der Figur
 DX = y-Koordinate des Punktes innerhalb der Figur

AH = 16hex

Rechteckige Fläche mit einem Muster ausfüllen (vergleiche »FillPattern«).
 CX = x1-Koordinate
 DX = y1-Koordinate
 DI = x2-Koordinate
 SI = y2-Koordinate

AH = 17hex

Füllmuster für die Funktion »16hex« definieren (vergleiche »Pattern«).
 CL = 1. Byte der 8x8 Punkte Matrix
 CH = 2. Byte der 8x8 Punkte Matrix
 DL = 3. Byte der 8x8 Punkte Matrix
 DH = 4. Byte der 8x8 Punkte Matrix
 DI = 5. und 6. Byte der 8x8 Punkte Matrix
 SI = 7. und 8. Byte der 8x8 Punkte Matrix

AH = 18hex

Begrenzungen des aktuellen Fensters definieren (vergleiche »GraphWindow«).
 CX = x-Koordinate der linken oberen Ecke
 DX = y-Koordinate der linken oberen Ecke
 DI = x-Koordinate der rechten unteren Ecke
 SI = y-Koordinate der rechten unteren Ecke

AH = 19hex

Aktuelles Fenster mit einem Bitmuster ausfüllen (vergleiche »FillWindow«).
 AL = Bitmuster

AH = 1Ahex

Aktuelles Fenster invertieren (vergleiche »InvertWindow«).

AH = 1Bhex

Aktuelles Fenster an eine andere Stelle kopieren (vergleiche »CopyWindow«).
 AL = neue Seite
 CX = neue x-Koordinate des Fensters
 DX = neue y-Koordinate des Fensters
 Die Assembler-Programme (Listing 1,2 und 4) wurden mit dem Microsoft Assembler »MASM« entwickelt. Sollten Sie diesen Assembler nicht besitzen, so können Sie die beiden Programme »hercules.com« und »hires.bin« auch hexadecimal eingeben (siehe Artikel »Geprüfte Eingabe«). Listing 3 zeigt das hexadezimale Listing von »hercules.com« und Listing 5 von »hires.bin«.

(Frank Brendle/sp)


```

[W12] 0: Page 60,132
[000] 1:
[LUC] 2: Title Grafik-System für Hercules-K
arte.
[000] 3:
[V38] 4: ; Funktions-Interrupts:
[ZKB] 5: Quit Equ 20H ; Pr
ogramm beenden.
[AKC] 6: DOS Equ 21H ; DO
S-Routinen aufrufen.
[RGF] 7: Resident Equ 27H ; Pr
ogramm resident machen.
[79G] 8: VideoInt Equ 60H ; Ne
ue Video-Routinen aufrufen.
[U5K] 9: GrafTabl Equ 1FH ; Ze
iger auf Zeichengenerator (128-255).
[000] 10:
[9A7] 11: ; Hardware-Interrupts:
[XKH] 12: Timer Equ 1CH ; In
terrupt, der durch den Timer aus-
[PS4] 13: löst wird. ; ge
[000] 14:
[EN9] 15: ; Adressen der Hercules-Karte:
[31H] 16: Index Equ 3B4H ; In
dex-Register der Hercules-Karte.
[S2L] 17: Control Equ 3B8H ; Co
ntrol-Register der Hercules-Karte.
[X2F] 18: Text Equ 28H ; We
rt um auf Text zu schalten.
[XHG] 19: Grafik Equ 10 ; We
rt um auf Grafik zu schalten.
[SCF] 20: Config Equ 3BFH ; Ko
nfigurations-Schalter.
[KFH] 21: Page1 Equ 0B000H ; St
artsegment der 1. Bildschirmseite.
[2GH] 22: Page2 Equ 0B800H ; St
artsegment der 2. Bildschirmseite.
[78G] 23: Length Equ (Page2-Page1)*8 ; Läng
e einer Seite in Worten.
[2TG] 24: XMax Equ 719 ; Ma
ximale Auflösung in X-Richtung.
[5SG] 25: YMax Equ 347 ; Ma
ximale Auflösung in Y-Richtung.
[PBE] 26: CharGen Equ 0FA6EH ; Ze
ichengenerator (0-127).
[18D] 27: BIOSSeg Equ 0F000H ; Se
gment des ROM-BIOS.
[000] 28:
[HW7] 29: ; Allgemeine Konstanten.
[NV8] 30: On Equ 255 ; We
rt für an.
[XM9] 31: Off Equ 0 ; We
rt für aus.
[VBH] 32: Blink Equ 5 ; Bl
inkgeschwindigkeit des Cursors.
[000] 33:
[EB4] 34: Code Segment;
[4M6] 35: Org 100H ; .C
OM-File.
[NA9] 36: Assume CS:Code,DS:Code,ES:Code
[000] 37:
[C24] 38: GSys Proc Near
[000] 39:
[FCF] 40: Start: Jmp Init ; Pr
ogramm installieren.
[000] 41:
[TFB] 42: ; Start der Video-Interrupt-Routin
e.
[PUE] 43: NInt: StI ; In
terrupts wieder zulassen.
[SCA] 44: Push BX ; Re
gister retten.
[6V1] 45: Push CX
[7V1] 46: Push DX
[9U1] 47: Push DI
[SU1] 48: Push SI
[XU1] 49: Push DS
[YU1] 50: Push ES
[SAF] 51: CID ; Be
i String-Befehlen aufwärts zählen.
[RCA] 52: Push CS ; CS
nach DS schaffen.
[8H1] 53: Pop DS
[Y9B] 54: Mov ES,Word Ptr [Page] ; Se
ite setzen.
[F2E] 55: Mov BL,Byte Ptr [Cursor] ; Cu
rsor-Modus retten.
[U7Y] 56: Mov Byte Ptr [CursSave],BL
[M4E] 57: Mov Byte Ptr [Cursor],Off ; Cu
rsor abschalten.
[R58] 58: Cmp Byte Ptr [CursMod],Off
[D82] 59: JZ NInt1
[0BB] 60: Call DrawCur ; Cu
rsor löschen.

```

```

[RXE] 61: NInt1: Cmp AH,(GTable-CmdTab)/2 ; Er
laubtes Kommando?
[ZHD] 62: JAE Exit ; We
nn nein --> Programm beenden.
[WH9] 63: Mov BL,AH ; Ko
mmando nach BL.
[MW9] 64: Mov BH,0 ; Au
f 16 Bit erweitern.
[99F] 65: ShL BX,1 ; Mi
t 2 multiplizieren und ausführen.
[TH8] 66: Call Word Ptr DS:[BX+CmdTab]
[ZRA] 67: Exit: Mov BL,Byte Ptr CS:[CursSave]
[93L] 68: Mov Byte Ptr CS:[Cursor],BL ;
Cursor-Modus wiederherstellen.
[L5D] 69: Pop ES ; Re
gister wiederherstellen.
[8H1] 70: Pop DS
[3H1] 71: Pop SI
[MG1] 72: Pop DI
[KH1] 73: Pop DX
[HH1] 74: Pop CX
[GH1] 75: Pop BX
[XR6] 76: IRet ; Fe
rtig!
[000] 77:
[YAD] 78: ; Hercules-Karte auf Text/Grafik s
chalten.
[NLD] 79: SetMode: Cmp AL,3 ; Gr
afik einschalten?
[ZP3] 80: JA SetModel
[FPD] 81: Mov AL,Text ; Te
xtmodus einschalten.
[YC6] 82: Mov SI,Offset TTable
[C98] 83: Mov Byte Ptr [Graphic],Off
[5E5] 84: Jmp Short SetMode2
[TPG] 85: SetModel: Mov AL,Grafik ; Gr
afikmodus einschalten.
[RB6] 86: Mov SI,Offset GTable
[GR7] 87: Mov Byte Ptr [Graphic],On
[ZSL] 88: SetMode2: Mov DX,Control ; Mo
dus im Control-Register setzen.
[DA2] 89: Out DX,AL
[24A] 90: Mov AH,0 ; 68
45 initialisieren.
[4CH] 91: Mov DL,Low Index ; Ad
resse des Index-Registers nach DX.
[2FF] 92: SetMode3: Mov AL,AH ; Re
gisternummer nach AL
[5L8] 93: Out DX,AL ; un
d ausgeben.
[NTG] 94: LodSB ; Da
zugehörenden Wert aus Tabelle holen
[2G1] 95: Inc DX
[5L8] 96: Out DX,AL ; un
d ausgeben.
[BF1] 97: Dec DX
[X6A] 98: Inc AH ; Nä
chstes Register.
[BLE] 99: Cmp AH,12 ; Al
le 12 Register initialisiert?
[723] 100: JnZ SetMode3
[SL0] 101: Ret
[000] 102:
[DX7] 103: ; Abprüfen, ob Grafik aktiv.
[3YH] 104: GetMode: Mov AL,Byte Ptr [Graphic] ; Mo
dus nach AL schaffen.
[SL0] 105: Ret
[000] 106:
[5A9] 107: ; Aussehen des Cursors setzen.
[TND] 108: SetCurs: Or DL,DL ; Cu
rsor abschalten?
[L44] 109: JnZ SetCurs1
[VFF] 110: Mov Byte Ptr [CursSave],Off; C
ursor ausschalten.
[SL0] 111: Ret
[41K] 112: SetCurs1: Mov Byte Ptr [CursSave],On; Cu
rsor anschalten.
[AF1] 113: Dec CX
[KDD] 114: Mov Byte Ptr [StartZ],CL ; We
rt abspeichern.
[P5D] 115: Sub DX,CX ; Hö
he des Cursors berechnen
[26D] 116: Mov Byte Ptr [Hoehe],DL ; un
d abspeichern.
[SL0] 117: Ret
[000] 118:
[3K8] 119: ; Cursor-Position setzen.
[BZH] 120: SetPos: Mov Byte Ptr [XPos],CL ; X-
Koordinate abspeichern.
[9NF] 121: Mov Byte Ptr [YPos],DL ; Y-
Koordinate abspeichern.
[SL0] 122: Ret
[000] 123:
[BTF] 124: ; Cursor-Position lesen (Rückgabe
in Register AX).

```



```
[FBG] 125: GetPos: Mov AL,Byte Ptr [XPos] ; X-
           Koordinate nach AL.
[PF] 126: Mov AH,Byte Ptr [YPos] ; Y-
           Koordinate nach AH.
[SLO] 127: Ret
[000] 128:
[YDD] 129: ; Zeichenmodus setzen (steht in Register AL).
[Y5F] 130: SetMod: Mov Byte Ptr [Mode],AL ; We
           rt abspeichern.
[SLO] 131: Ret
[000] 132:
[AWB] 133: ; Anzeigeseite wählen (Seite steht in AL).
[M7G] 134: Disp: Cmp Byte Ptr [Graphic],Off; Grafikmodus aktiv?
[6D2] 135: JZ DispE
[CME] 136: Mov DX,Control ; Bit
           shen Wert einlesen.
[576] 137: Dec AL ; Seite
           ite 17
[SG2] 138: Jnz Displ
[UZF] 139: Mov AL,Grafik ; Ob
           erstes Bit gelöscht --> Seite 1.
[5RF] 140: DB 0B9H ; Nä
           chsten Befehl unschädlich machen.
[HNH] 141: Displ: Or AL,Grafik+128 ; Ob
           erstes Bit gesetzt --> Seite 2.
[F19] 142: Out DX,AL ; Wert
           rt ausgeben.
[7W2] 143: DispE: Ret
[000] 144:
[9VC] 145: ; Seite, die beschrieben wird, auswählen.
[FEH] 146: SetPage: Mov AH,High Page1 ; Wert
           rt für 1. Seite aufsetzen.
[576] 147: Dec AL ; Seite
           ite 17
[9R3] 148: JZ SetPage1
[0EB] 149: Mov AH,High Page2 ; Wert
           rt für 2. Seite.
[YPF] 150: SetPage: Mov Byte Ptr [Page+1],AH ; Wert
           rt abspeichern.
[SLO] 151: Ret
[000] 152:
[52F] 153: ; Farbe eines Punktes ermitteln (Rückgabe in AL).
[HOL] 154: GetCol: Add CX,Word Ptr [StartX] ; Fenster
           berücksichtigen.
[XB7] 155: Mov AX,Word Ptr [StartY]
[K92] 156: Add AX,DX
[BT] 157: Call Zoff ; Zeilen-Offset berechnen.
[HVD] 158: Mov DI,CX ; X-Koordinate nach DI schaffen.
[F9C] 159: Shr DI,1 ; X-Koordinate durch 8 teilen
[RR1] 160: Shr DI,1
[RR1] 161: Shr DI,1
[F9G] 162: Add DI,AX ; un-
           d Zeilen-Offset addieren --> Adresse.
[ZVA] 163: And CL,7 ; Bit
           tmuster berechnen.
[SD2] 164: Mov AL,128
[Y52] 165: Shr AL,CL
[S5C] 166: And AL,Byte Ptr ES:[DI] ; Punkt
           nkt abprüfen.
[DH3] 167: JZ GetColE
[M29] 168: Mov AL,1 ; Punkt
           nkt war gesetzt.
[KE3] 169: GetColE: Ret
[000] 170:
[TMG] 171: ; Punkt setzen (CX enthält die X-, DX die Y-Koordinate).
[GRK] 172: Point: Add CX,Word Ptr [StartX] ; Fenster
           berücksichtigen.
[897] 173: Add DX,Word Ptr [StartY]
[90K] 174: Point1: Cmp CX,Word Ptr [StartX] ; X-Koordinate zu klein?
[HZ2] 175: JB PointE
[5EF] 176: Cmp DX,Word Ptr [StartY] ; Y-Koordinate zu klein?
[HZ2] 177: JB PointE
[ZRE] 178: Cmp CX,Word Ptr [EndX] ; X-Koordinate zu groß?
[FZ2] 179: JA PointE
[ZSE] 180: Cmp DX,Word Ptr [EndY] ; Y-Koordinate zu groß?
[FZ2] 181: JA PointE
[UGC] 182: Mov AX,DX ; Zeilen-Offset berechnen.
[VF2] 183: Call Zoff
[HVD] 184: Mov DI,CX ; X-Koordinate nach DI schaffen.
```

```
[F9C] 185: Shr DI,1 ; X-Koordinate durch 8 teilen
[RR1] 186: Shr DI,1
[RR1] 187: Shr DI,1
[F9G] 188: Add DI,AX ; un-
           d Zeilen-Offset addieren --> Adresse.
[ZVA] 189: And CL,7 ; Bit
           tmuster berechnen.
[SD2] 190: Mov AL,128
[Y52] 191: Shr AL,CL
[93G] 192: Mov AH,Byte Ptr [Mode] ; Zeichen-Modus überprüfen.
[MA8] 193: Or AH,AH ; Punkt
           nkt setzen?
[543] 194: Jnz Point2
[9NA] 195: Or Byte Ptr ES:[DI],AL ; Punkt
           nkt setzen.
[SLO] 196: Ret
[LGB] 197: Point2: Dec AH ; Punkt
           nkt löschen?
[B43] 198: Jnz Point3
[1H1] 199: Not AL
[S9B] 200: And Byte Ptr ES:[DI],AL ; Punkt
           nkt löschen.
[SLO] 201: Ret
[EXE] 202: Point3: Xor Byte Ptr ES:[DI],AL ; Punkt
           nkt invertieren.
[WA3] 203: PointE: Ret
[000] 204:
[VNF] 205: ; Linie von X1,Y1 nach X2,Y2 ziehe
           n (X1=CX, Y1=DX, X2=DI, Y2=SI).
[D8K] 206: Line: Add CX,Word Ptr [StartX] ; Fenster
           berücksichtigen.
[787] 207: Add DI,Word Ptr [StartX]
[897] 208: Add DX,Word Ptr [StartY]
[T87] 209: Add SI,Word Ptr [StartY]
[6SD] 210: Cmp DI,CX ; X1
           soll kleiner als X2 werden.
[CA2] 211: JAE Line1
[YKA] 212: Xchg DI,CX ; X-
           Werte tauschen.
[KLA] 213: Xchg SI,DX ; Y-
           Werte tauschen.
[P4H] 214: Line1: Mov Word Ptr [X1],CX ; X-
           Werte zwischenspeichern.
[FL5] 215: Mov Word Ptr [X2],DI
[T2F] 216: Mov Word Ptr [Richtung],0 ; Variable
           vorbesetzen.
[6T9] 217: Sub DI,CX ; DeltaX
           berechnen.
[A37] 218: Mov Word Ptr [DeltaX],DI
[AF2] 219: Jnz Line2
[TS7] 220: Mov Word Ptr [Richtung],-1
[9PG] 221: Line2: Mov Word Ptr [StepY],1 ; StepY
           mit 1 vorbesetzen.
[4XB] 222: Cmp SI,DX ; Wenn
           Y2>Y1 bleibt StepY=1
[772] 223: JA Line3
[ELB] 224: Mov Word Ptr [StepY],-1 ; Schritt
           StepY=-1.
[65H] 225: Line3: Mov Word Ptr [Y1],DX ; Y-
           Werte zwischenspeichern.
[VP5] 226: Mov Word Ptr [Y2],SI
[ZT9] 227: Sub DX,SI ; DeltaY
           berechnen.
[ZE2] 228: Jns Line4
[EE9] 229: Neg DX ; Betrag
           berechnen.
[189] 230: Line4: Mov Word Ptr [DeltaY],DX
[UWR] 231: Line5: Rol Word Ptr [LineStyle],1 ; Schritt
           ll der Punkt überhaupt gezeichnet werden?
[A6B] 232: Jnc Line6 ; Punkt
           nkt nicht zeichnen!
[45C] 233: Mov CX,Word Ptr [X1] ; Punkt
           nkt zeichnen.
[PZ5] 234: Mov DX,Word Ptr [Y1]
[4E3] 235: Call Point1
[DUG] 236: Line6: Cmp Word Ptr [Richtung],0 ; Prüfen,
           ob Richtung < 0.
[EF2] 237: Jns Line7
[R9E] 238: Mov AX,Word Ptr [Y2] ; Prüfen,
           ob Y2 erreicht ist.
[LY5] 239: Cmp AX,Word Ptr [Y1]
[Z57] 240: JZ LineE ; Fertig.
[V8D] 241: Mov AX,Word Ptr [StepY] ; Nächster
           Punkt.
[1G5] 242: Add Word Ptr [Y1],AX
[L47] 243: Mov AX,Word Ptr [DeltaX]
[F08] 244: Add Word Ptr [Richtung],AX
[V24] 245: Jmp Short Line5
[ALG] 246: Line7: Mov AX,Word Ptr [X2] ; Prüfen,
           ob X2 erreicht ist.
[DY5] 247: Cmp AX,Word Ptr [X1]
```



```

[Z57] 248: JZ LineE ; Fe
[6RB] 249: Inc Word Ptr [X1] ; Nä
        chster Punkt.
[P47] 250: Mov AX,Word Ptr [DeltaY]
[X18] 251: Sub Word Ptr [Richtung],AX
[V24] 252: Jmp Short Line5
[ZU8] 253: LineE: Ret ; Fe

[000] 254: ; Aussehen der Linien definieren (
[8BF] 255: Bitmuster in CX).
[DGL] 256: SetStyle: Mov Word Ptr [LineStyle],CX; Bi
        tmuster abspeichern.
[SL0] 257: Ret
[000] 258:
[DAH] 259: ; Window definieren (X1, Y1, X2, Y
        2 wie immer in CX, DX, DI, SI).
[CZF] 260: DefWin: Mov Word Ptr [StartX],CX ; We
        rte abspeichern.
[R07] 261: Mov Word Ptr DS:[EndX],DI
[HA7] 262: Mov Word Ptr [StartY],DX
[BF6] 263: Mov Word Ptr [EndY],SI
[SL0] 264: Ret
[000] 265:
[AHA] 266: ; Window mit Bitmuster in AL fülle
        n.
[7CH] 267: FilWin: Std ; Be
        im Stringbefehl abwärtszählen.
[E8E] 268: Mov DL,AL ; Bi
        tmuster zwischen speichern.
[KRG] 269: Call BitM ; Bi
        tmuster für Byte-Grenzen berechnen.
[MCF] 270: Mov AX,Word Ptr [StartY] ; AX
        auf 1. Zeile setzen.
[PDG] 271: Mov BX,Word Ptr [StartX] ; X-
        Offset berechnen (StartX).
[H8F] 272: Mov DI,Word Ptr [EndX] ; X-
        Offset berechnen (EndX).
[67E] 273: Shr BX,1 ; X-
        Koordinaten durch 8 dividieren.
[LS1] 274: Shr BX,1
[LS1] 275: Shr BX,1
[RR1] 276: Shr DI,1
[RR1] 277: Shr DI,1
[RR1] 278: Shr DI,1
[VTD] 279: FilWin1: Push AX ; Ze
        ilennummer retten.
[TV9] 280: Push BX ; X-
        Werte retten.
[9U1] 281: Push DI
[BTC] 282: Call Zoff ; Ze
        ilen-Offset berechnen.
[MRA] 283: Add BX,AX ; Ad
        ressen berechnen.
[B82] 284: Add DI,AX
[91C] 285: Not CX ; Bi
        tmuster invertieren.
[WXA] 286: And Byte Ptr ES:[BX],CL ; Bi
        ts löschen
[C26] 287: And Byte Ptr ES:[DI],CH
[XZA] 288: Mov AL,DL ; un
        d neu beschreiben.
[7FD] 289: Not CX ; Bi
        tmuster wiederherstellen.
[E72] 290: And AL,CL
[VZ5] 291: Or Byte Ptr ES:[BX],AL
[P92] 292: Mov AL,DL
[U62] 293: And AL,CH
[YW5] 294: Or Byte Ptr ES:[DI],AL
[TMC] 295: Mov AL,DL ; Bi
        tmuster nach AL holen.
[DX9] 296: Dec DI ; DI
        dekrementieren.
[OEB] 297: ; Bytes zwischen den Grenzen fülle
        n.
[5RC] 298: FilWin2: Cmp BX,DI ; Al
        le Bytes gefüllt?
[XZC] 299: JZ FilWin3 ; We
        nn ja --> nächste Zeile.
[SDA] 300: StoSB ; Bi
        tmuster ablegen.
[30B] 301: Jmp Short FilWin2 ; Nä
        chstes Byte.
[EUB] 302: FilWin3: Pop DI ; X-
        Werte holen.
[GH1] 303: Pop BX
[NMC] 304: Pop AX ; Ze
        ilennummer wieder holen.
[XN8] 305: Inc AX ; Nä
        chste Zeile
[W2A] 306: Cmp AX,Word Ptr [EndY] ; Fe
        rtig?
[TK3] 307: JnA FilWin1

```

```

[9G6] 308: Ret ; Fe
[000] 309: rtig!
[T7F] 310: ; Zeichen fortlaufend schreiben (Z
        eichen in AL).
[9MB] 311: Char: Cmp AL,13 ; Wa
        genrücklauf?
[7F2] 312: Jnz Char1
[YCE] 313: Mov Byte Ptr [XPos],0 ; Cu
        rsor an linken Rand.
[SL0] 314: Ret
[S5F] 315: Char1: Mov CX,Word Ptr [XPos] ; X-
        Koordinate holen.
[9D7] 316: Cmp AL,8 ; Ba
        ckspace?
[CF2] 317: Jnz Char2
[EUA] 318: Or CL,CL ; Li
        nker Rand erreicht?
[NB2] 319: JZ CharE
[57D] 320: Dec Byte Ptr [XPos] ; Sc
        hritt nach links
[PU2] 321: CharE: Ret
[5FE] 322: Char2: ShL CX,1 ; un
        d mit 8 multiplizieren.
[2S1] 323: ShL CX,1
[2S1] 324: ShL CX,1
[TXG] 325: Add CX,Word Ptr [StartX] ; Fe
        nster berücksichtigen.
[PID] 326: Mov DX,Word Ptr [YPos] ; Y-
        Koordinate holen
[CRC] 327: ShL DX,1 ; un
        d mit 8 multiplizieren.
[3S1] 328: ShL DX,1
[3S1] 329: ShL DX,1
[XXG] 330: Add DX,Word Ptr [StartY] ; Fe
        nster berücksichtigen.
[0Z9] 331: Cmp AL,10 ; Ze
        ilenvorschub?
[092] 332: JZ Char4
[4MA] 333: Push CX ; Ko
        ordinaten retten.
[7V1] 334: Push DX
[3YB] 335: Call DrawChr ; Ze
        ichen ausgeben.
[B4E] 336: Pop DX ; Ko
        ordinaten wiederherstellen.
[HH1] 337: Pop CX
[34F] 338: Add CX,16 ; Pr
        üfen, ob Ende der Zeile erreicht.
[UG6] 339: Cmp CX,Word Ptr [EndX]
[ERC] 340: JA Char3 ; En
        de der Zeile ist erreicht!
[PBG] 341: Inc Byte Ptr [XPos] ; X-
        Koordinate weiterschalten.
[4L6] 342: Ret ; Fe
        rtig.
[ESG] 343: Char3: Mov Byte Ptr [XPos],0 ; Wi
        eder an linken Rand.
[L1E] 344: Char4: Add DX,16 ; Un
        terer Rand erreicht?
[WG6] 345: Cmp DX,Word Ptr [EndY]
[L72] 346: JA Char5
[KVF] 347: Inc Byte Ptr [YPos] ; Cu
        rsor eine Zeile nach unten.
[SL0] 348: Ret
[ZWE] 349: Char5: Mov DL,8 ; 8
        Zeilen nach oben scrollen.
[5F4] 350: Jmp Short ScrUp1
[000] 351:
[YVT] 352: ; Zeichenkette auf dem Bildschirm
        ausgeben (Zeiger in DS:SI, 1. Byte=Länge des
        Strings).
[PRK] 353: Print: Add CX,Word Ptr [StartX] ; Fe
        nster berücksichtigen.
[897] 354: Add DX,Word Ptr [StartY]
[5ZA] 355: Add DX,8 ; Y-
        Koordinate erlaubt?
[WG6] 356: Cmp DX,Word Ptr [EndY]
[NZ2] 357: JA PrintE
[1GE] 358: Sub DX,8 ; Y-
        Koordinate wiederherstellen.
[CLA] 359: Mov BX,SP ; SP
        nach BX schaffen.
[URC] 360: Mov DS,Word Ptr SS:[BX+4] ; DS
        vom Stack holen.
[9EC] 361: LodSB ; LÄ
        nge des Strings holen
[NMC] 362: Mov AH,AL ; un
        d Zähler in AH aufsetzen.
[HAH] 363: Print1: Add CX,8 ; Li
        egt das Zeichen noch im Fenster?
[P77] 364: Cmp CX,Word Ptr CS:[EndX]
[6H9] 365: JA PrintE ; We
        nn nein --> Ende.

```


[FTF]	366:	Sub	CX,8	: X-
[M39]	367:	LodSB		: Ze
[RCA]	368:	Push	AX	: Re
[6V1]	369:	Push	CX	
[7V1]	370:	Push	DX	
[SU1]	371:	Push	SI	
[XU1]	372:	Push	DS	
[D3C]	373:	Call	DrawChr	: Ze
[K5D]	374:	Pop	DS	: Re
[3H1]	375:	Pop	SI	
[KH1]	376:	Pop	DX	
[HH1]	377:	Pop	CX	
[FH1]	378:	Pop	AX	
[R1F]	379:	Add	CX,8	: Ei
[HBB]	380:	Dec	AH	: Zä
[KZA]	381:	Jnz	Print1	: Nä
[2B3]	382:	PrintE:	Ret	
[000]	383:			
[LOF]	384:		Fenster nach oben scrollen (Zahl	
[NWH]	385:	ScrUp:	Mov DL,AL	: Za
[FDE]	386:	ScrUp1:	Mov DH,0	: un
[SCB]	387:	Call	BitM	: Bi
[5ZD]	388:	Mov	BX,Word Ptr [StartX]	: X-
[UH9]	389:	ShR	BX,1	: du
[LS1]	390:	ShR	BX,1	
[LS1]	391:	ShR	BX,1	
[M4C]	392:	Mov	Word Ptr [X1],BX	: un
[E9F]	393:	Mov	AX,Word Ptr [StartY]	: Nu
[Y2F]	394:	Mov	SI,AX	: We
[EX8]	395:	Add	SI,DX	: bu
[5UA]	396:	Push	DX	: Ve
[SRE]	397:	Mov	DX,Word Ptr [EndX]	: 2.
[6P9]	398:	ShR	DX,1	: du
[NS1]	399:	ShR	DX,1	
[NS1]	400:	ShR	DX,1	
[3KF]	401:	Sub	DX,BX	: Di
[2FD]	402:	ScrUp2:	Push AX	: Y1
[N2D]	403:	Call	Zoff	: Ze
[SH9]	404:	Mov	DI,AX	: na
[ZCF]	405:	Add	DI,Word Ptr [X1]	: X-
[RH7]	406:	Mov	AX,SI	: Y2
[N2D]	407:	Call	Zoff	: Ze
[HN9]	408:	Mov	BX,AX	: na
[VDF]	409:	Add	BX,Word Ptr [X1]	: X-
[MYG]	410:		Bits bis zur 1. Byte-Grenze vers	
[B4E]	411:	Not	CX	: Bi
[S26]	412:	And	Byte Ptr ES:[DI],CL	
[VH1]	413:	Not	CX	
[FDH]	414:	Mov	AL,Byte Ptr ES:[BX]	: Bi
[PL7]	415:	And	AL,CL	: ma
[N8C]	416:	Or	Byte Ptr ES:[DI],AL	: un
[G8H]	417:		Bits bis zur 1. Byte-Grenze vers	
[3CB]	418:	Add	BX,DX	: By
[R82]	419:	Add	DI,DX	
[B4E]	420:	Not	CX	: Bi
[C26]	421:	And	Byte Ptr ES:[DI],CH	

[VH1]	422:	Not	CX	
[FDH]	423:	Mov	AL,Byte Ptr ES:[BX]	: Bi
[2L7]	424:	And	AL,CH	: ma
[N8C]	425:	Or	Byte Ptr ES:[DI],AL	: un
[N3F]	426:		Bytes zwischen diesen Grenzen ve	
[OV8]	427:	Mov	DH,DL	: Zä
[YEC]	428:	Dec	DI	: Ad
[E6F]	429:	StD		: Be
[A04]	430:	ScrUp3:	Dec BX	
[X3A]	431:	Mov	AL,Byte Ptr ES:[BX]	: By
[N29]	432:	StoSB		: un
[ES6]	433:	Dec	DH	: Fe
[VV9]	434:	Jnz	ScrUp3	: Nä
[BMA]	435:	Pop	AX	: AX
[HY8]	436:	Inc	AX	: Nä
[LF1]	437:	Inc	SI	
[G2A]	438:	Cmp	SI,Word Ptr [EndY]	: Fe
[8Y2]	439:	JnA	ScrUp2	
[EBA]	440:		Freigewordene Zeilen löschen.	
[48G]	441:	Pop	SI	: Ve
[F3H]	442:	Mov	AX,Word Ptr [EndY]	: Ze
[FR7]	443:	Sub	AX,SI	: be
[CKD]	444:	ScrUp4:	Push AX	: Ze
[BTC]	445:	Call	Zoff	: Ze
[9UA]	446:	Mov	DI,AX	: un
[ZCF]	447:	Add	DI,Word Ptr [X1]	: X-
[V3F]	448:	Cmp	Byte Ptr CS:[Model],1	: Ne
[513]	449:	Jnz	ScrUp5	
[07C]	450:		Bits bis zur linken Byte-Grenze	
[P5A]	451:	Or	Byte Ptr ES:[DI],CL	: Bi
[YGC]	452:		Bits bis zur rechten Byte-Grenze	
[R82]	453:	Add	DI,DX	
[PW5]	454:	Or	Byte Ptr ES:[DI],CH	
[V29]	455:		Bytes dazwischen löschen:	
[M2E]	456:	Mov	AL,255	: Bi
[TF4]	457:	Jmp	Short ScrUp6	
[07C]	458:		Bits bis zur linken Byte-Grenze	
[7SE]	459:	ScrUp5:	Not CX	: Bi
[52B]	460:	And	Byte Ptr ES:[DI],CL	: Bi
[7UC]	461:		Bits bis zur rechten Byte-Grenze	
[R82]	462:	Add	DI,DX	
[C26]	463:	And	Byte Ptr ES:[DI],CH	
[V29]	464:		Bytes dazwischen löschen:	
[7FD]	465:	Not	CX	: Bi
[5SE]	466:	Mov	AL,0	: Bi
[TWC]	467:	ScrUp6:	Push CX	: Bi
[KCB]	468:	Mov	CX,DX	: Za
[SCA]	469:	Dec	CX	: Za
[79B]	470:	Dec	DI	: Ze
[X79]	471:	Rep	StoSB	: By
[XED]	472:	Pop	CX	: Bi
[ELE]	473:	Pop	AX	: Ze
[HY8]	474:	Inc	AX	: Nä
[W2A]	475:	Cmp	AX,Word Ptr [EndY]	: Fe
			rtig?	


```

[MY2] 476: JnA ScrUp4
[SL0] 477: Ret
[000] 478:
[KRH] 479: ; Zeichenkette auf dem Bildschirm
[83H] 480: ausgeben (Zeiger in DS:SI,
; 1. Byte=Länge des Strings, CX=X,
DX=Y, DI=Vergrößerung).
[WYK] 481: PrintB: Add CX,Word Ptr [StartX] ; Fe
nster berücksichtigen.
[897] 482: Add DX,Word Ptr [StartY]
[75F] 483: Mov BX,DI ; Ve
rgrößerung nach BX schaffen.
[V82] 484: Mov BL,BH
[3V1] 485: Mov BH,0
[7GC] 486: ShL BX,1 ; Mi
t 8 multiplizieren --> Höhe des Zeichens.
[1S1] 487: ShL BX,1
[1S1] 488: ShL BX,1
[59B] 489: Add DX,BX ; Y-
Koordinate erlaubt?
[WG6] 490: Cmp DX,Word Ptr [EndY]
[6E3] 491: JA PrintBE
[1UE] 492: Sub DX,BX ; Y-
Koordinate wiederherstellen.
[CLA] 493: Mov BX,SP ; SP
nach BX schaffen.
[URC] 494: Mov DS,Word Ptr SS:[BX+4] ; DS
vom Stack holen.
[9EC] 495: LodSB ; LÄ
nge des Strings holen
[NMC] 496: Mov AH,AL ; un
d Zähler in AH aufsetzen.
[N3G] 497: PrintB1: Mov BX,DI ; Hö
he des Zeichens berechnen.
[3V1] 498: Mov BH,0
[1S1] 499: ShL BX,1
[1S1] 500: ShL BX,1
[1S1] 501: ShL BX,1
[SXE] 502: Add CX,BX ; Li
egt das Zeichen noch im Fenster?
[P77] 503: Cmp CX,Word Ptr CS:[EndX]
[RS9] 504: JA PrintBE ; We
nn nein --> Ende.
[F3G] 505: Sub CX,BX ; X-
Koordinate wieder zurückschalten.
[SY7] 506: Push BX ; We
rt retten.
[T2C] 507: Mov BX,DI ; Ve
rgrößerung nach BX.
[M39] 508: LodSB ; Ze
ichen holen.
[RCA] 509: Push AX ; Re
gister retten.
[6V1] 510: Push CX
[7V1] 511: Push DX
[9U1] 512: Push DI
[SU1] 513: Push SI
[XU1] 514: Push DS
[KKB] 515: Call BigChr ; Ze
ichen schreiben.
[K5D] 516: Pop DS ; Re
gister wiederherstellen.
[3H1] 517: Pop SI
[MG1] 518: Pop DI
[KH1] 519: Pop DX
[HH1] 520: Pop CX
[FH1] 521: Pop AX
[CKE] 522: Pop BX ; X-
Vergrößerung vom Stack holen.
[RBF] 523: Add CX,BX ; Ei
ne Zeichenposition nach rechts.
[HBB] 524: Dec AH ; Zä
hler dekrementieren.
[F6B] 525: JnZ PrintB1 ; Nä
chstes Zeichen.
[XC3] 526: PrintBE: Ret
[000] 527:
[6LA] 528: ; Beliebig umrandete Figur füllen.
[UBL] 529: FillShp: Add CX,Word Ptr [StartX] ; Fe
nster berücksichtigen.
[897] 530: Add DX,Word Ptr [StartY]
[4HL] 531: Mov SI,Offset Init ; SI
mit Anfangsadresse des 1. Puffers laden.
[2XL] 532: Mov DI,Offset Buffer2 ; DI
mit Anfangsadresse des 2. Puffers laden.
[2VG] 533: Mov Word Ptr [SI],CX ; X-
Koordinate im Puffer ablegen.
[RYG] 534: Mov Word Ptr [SI+2],DX ; Y-
Koordinate im Puffer ablegen.
[PEB] 535: Add SI,4 ; Ze
iger weiterschalten.
[MZK] 536: FillShp1: Dec SI ; Ze
iger auf letzte Eintragung setzen.
[WE1] 537: Dec SI

```

```

[VWC] 538: Mov AX,Word Ptr [SI] ; Y-
Koordinate holen.
[WE1] 539: Dec SI
[WE1] 540: Dec SI
[WVC] 541: Mov CX,Word Ptr [SI] ; X-
Koordinate holen.
[C5A] 542: Dec CX ; Sc
hritt nach links.
[T6E] 543: Call Test ; Pu
nkt prüfen und evtl. merken.
[HKA] 544: Inc CX ; Sc
hritt nach rechts.
[1G1] 545: Inc CX
[OM2] 546: Call Test
[WV9] 547: Dec CX ; Sc
hritt nach oben.
[8F1] 548: Dec AX
[OM2] 549: Call Test
[A8A] 550: Inc AX ; Sc
hritt nach unten.
[ZF1] 551: Inc AX
[OM2] 552: Call Test
[D6D] 553: Cmp SI,Offset Init ; Pu
ffer abgearbeitet?
[9U3] 554: JZ FillShp2
[YF6] 555: Cmp SI,Offset Buffer2
[Z04] 556: JnZ FillShp3
[1CD] 557: FillShp2: Mov SI,DI ; Pu
ffer tauschen.
[PRF] 558: Cmp DI,Offset Buffer2 ; Wu
rde vorher Puffer 2 belegt?
[TS3] 559: JB FillShp3
[BSK] 560: Mov DI,Offset Init ; DI
enthält Startadresse von Puffer 1.
[ZC5] 561: Jmp Short FillShp4
[PTN] 562: FillShp3: Mov DI,Offset Buffer2 ; DI
enthält Startadresse von Puffer 2.
[WGF] 563: FillShp4: Cmp SI,Offset Init ; Sc
hon ganz fertig?
[9U3] 564: JZ FillShpE
[YF6] 565: Cmp SI,Offset Buffer2
[85A] 566: JnZ FillShp1 ; We
itermachen.
[8S3] 567: FillShpE: Ret
[000] 568:
[GYL] 569: ; Prüfen, ob ein Punkt gesetzt ist
und diesen evtl. abspeichern.
[NNC] 570: Test: Push AX ; Ko
ordinaten retten.
[6V1] 571: Push CX
[7GE] 572: Cmp AX,Word Ptr [StartY] ; Ob
erer Rand erreicht?
[EC2] 573: JB TestE
[NVE] 574: Cmp AX,Word Ptr [EndY] ; Un
terer Rand erreicht?
[CC2] 575: JA TestE
[GGE] 576: Cmp CX,Word Ptr [StartX] ; Li
nker Rand erreicht?
[EC2] 577: JB TestE
[4VE] 578: Cmp CX,Word Ptr [EndX] ; Re
chter Rand erreicht?
[CC2] 579: JA TestE
[BTC] 580: Call Zoff ; Ze
ilen-Offset berechnen.
[36C] 581: Mov BX,CX ; X-
Offset in BX aufsetzen.
[LS1] 582: ShR BX,1
[LS1] 583: ShR BX,1
[LS1] 584: ShR BX,1
[9EF] 585: Add BX,AX ; Ko
mplette Adresse steht nun in BX.
[WHA] 586: Mov AL,128 ; Bi
tmaske berechnen
[4U1] 587: And CL,7
[YA2] 588: ShR AL,C1
[C8D] 589: Mov AH,AL ; un
d in AH zwischenspeichern.
[N7B] 590: Cmp Byte Ptr [Mode],1 ; Mo
dus prüfen.
[7CG] 591: JZ Test1 ; Bi
ldpunkte sollen gelöscht werden.
[M6C] 592: And AL,Byte Ptr ES:[BX] ; Pu
nkt abprüfen.
[OLA] 593: JnZ TestE ; Pu
nkt war gesetzt!
[URA] 594: Or Byte Ptr ES:[BX],AH ; Pu
nkt setzen.
[M3D] 595: Pop CX ; Ko
ordinaten vom Stack holen.
[FH1] 596: Pop AX
[V3F] 597: Mov Word Ptr DS:[DI],CX ; X-
Koordinate abspeichern.
[4F1] 598: Inc DI
[4F1] 599: Inc DI
[S3F] 600: Mov Word Ptr DS:[DI],AX ; Y-

```



```

Koordinate abspeichern.
[4F1] 601: Inc DI
[4F1] 602: Inc DI
[SLO] 603: Ret
[SHE] 604: Test1: And AL,Byte Ptr ES:[BX] ; Pu
nkt abprüfen.
[U8D] 605: JZ TestE ; Pu
nkt war schon gelöscht!
[9CB] 607: And AH,Byte Ptr ES:[BX],AH ; Pu
nkt löschen.
[M3D] 608: Pop CX ; Ko
ordinaten vom Stack holen.
[FH1] 609: Pop AX
[V3F] 610: Mov Word Ptr DS:[DI],CX ; X-
Koordinate abspeichern.
[4F1] 611: Inc DI
[4F1] 612: Inc DI
[S3F] 613: Mov Word Ptr DS:[DI],AX ; Y-
Koordinate abspeichern.
[4F1] 614: Inc DI
[4F1] 615: Inc DI
[SLO] 616: Ret
[F9F] 617: TestE: Pop CX ; Ko
ordinaten vom Stack holen.
[FH1] 618: Pop AX
[9G6] 619: Ret ; Fe
rtig!
[000] 620:
[FTB] 621: ; Gesamten Bildschirminhalt lösche
n.
[K5F] 622: Clear: XOr AX,AX ; Bi
tmuster in AX aufsetzen.
[SEF] 623: Cmp Byte Ptr DS:[Mode],1 ; Bi
ldsschirm heil machen?
[213] 624: JnZ Clear1
[K56] 625: Not AX ; AX
=OFFFHH
[4CK] 626: Clear1: Mov CX,Length ; LÄ
nge des Bildschirmspeichers.
[LNB] 627: XOr DI,DI ; An
fangsadresse nach DI.
[L6B] 628: Rep StoSW ; Bi
ldschrift löschen.
[SLO] 629: Ret
[000] 630:
[TES] 631: ; Bildschirminhalt der aktivierten
Seite in die nicht-aktivierte kopieren.
[S2M] 632: ScrCopy: Mov AX,ES ; Se
gment der
[RB2] 633: Mov DS,AX
[36K] 634: XOr AH,8 ; Se
gment der nicht-aktivierten Seite nach ES.
[SB2] 635: Mov ES,AX
[63D] 636: Mov CX,Length ; ZÄ
hler in CX aufsetzen.
[SR9] 637: XOr DI,DI ; DI
und SI löschen.
[E92] 638: XOr SI,SI
[6GE] 639: Rep MovSW ; Bi
ldschriftinhalt übertragen.
[SLO] 640: Ret
[000] 641:
[PHF] 642: ; Bildschirminhalt der beiden Seit
en vertauschen.
[5GM] 643: Swap: Mov AX,ES ; Se
gment der nicht-aktivierten Seite nach DS.
[1U1] 644: XOr AH,8
[RB2] 645: Mov DS,AX
[63D] 646: Mov CX,Length ; ZÄ
hler in CX aufsetzen.
[SR9] 647: XOr DI,DI ; DI
und SI löschen.
[E92] 648: XOr SI,SI
[GVL] 649: Swapl: LodSW ; Wo
rt aus nicht-aktivierter Seite holen.
[AXH] 650: Mov BX,Word Ptr ES:[DI] ; Wo
rt aus aktivierter Seite holen.
[Z5E] 651: StoSW ; We
rte vertauscht abspeichern.
[5W6] 652: Mov Word Ptr DS:[SI-2],BX
[E1A] 653: Loop Swapl ; NÄ
chste Worte.
[SLO] 654: Ret
[000] 655:
[BD7] 656: ; Fenster invertieren.
[5SE] 657: Invert: Call BitM ; Bi
tmuster berechnen.
[6AE] 658: Mov BX,Word Ptr [StartX] ; X-
Offsets berechnen:
[4N9] 659: Shr BX,1 ; Du
rch 8 dividieren.
[LS1] 660: Shr BX,1
[LS1] 661: Shr BX,1
[CAF] 662: Mov DI,Word Ptr [EndX] ; Da
sselbe mit EndX machen.

```

```

[RR1] 663: Shr DI,1
[RR1] 664: Shr DI,1
[RR1] 665: Shr DI,1
[AKH] 666: Mov AX,Word Ptr [StartY] ; Ze
ilennummer in AX aufsetzen.
[43D] 667: Invert1: Push AX ; Re
gister retten.
[5V1] 668: Push BX
[9U1] 669: Push DI
[BTC] 670: Call Zoff ; Ze
ilen-Offset berechnen.
[9FC] 671: Add BX,AX ; By
te-Adresse des 1. Punktes.
[D8F] 672: Add DI,AX ; By
te-Adresse des letzten Punktes.
[X1F] 673: XOr Byte Ptr ES:[BX],CL ; Li
nken Rand invertieren.
[U9F] 674: XOr Byte Ptr ES:[DI],CH ; Re
chten Rand invertieren.
[4UG] 675: Invert2: Dec DI ; By
tes dazwischen invertieren.
[A87] 676: Cmp DI,BX ; Fe
rtig?
[GM3] 677: JZ Invert3
[1S5] 678: Not Byte Ptr ES:[DI]
[H3B] 679: Jmp Short Invert2 ; NÄ
chstes Byte.
[OYF] 680: Invert3: Pop DI ; Re
gister wiederherstellen.
[GH1] 681: Pop BX
[FH1] 682: Pop AX
[50B] 683: Inc AX ; Ze
ilennummer erhöhen.
[W2A] 684: Cmp AX,Word Ptr [EndY] ; Fe
rtig?
[ZDA] 685: JnA Invert1 ; NÄ
chste Zeile.
[SLO] 686: Ret
[000] 687:
[ZDG] 688: ; Fenster kopieren (Neue X1/Y1-Koo
rdinaten in CX und DX).
[W5D] 689: Copy: Push AX ; Se
itennummer retten.
[USH] 690: Cmp DX,Word Ptr [StartY] ; Li
egt das neue Fenster tiefer als
[AB7] 691: JA Copy1 ; da
s alte?
[GFC] 692: Mov Word Ptr [StepY],1 ; Sc
hrittweite=1.
[6W7] 693: Mov Word Ptr [Y1],DX ; Y1
=X1.
[RWA] 694: Mov AX,Word Ptr [StartY] ; Y2
=StartY.
[XK5] 695: Mov Word Ptr [Y2],AX
[R4G] 696: Mov BX,Word Ptr [EndY] ; Hö
he des Fensters berechnen
[NA2] 697: Sub BX,AX
[RCD] 698: Mov Word Ptr [DeltaY],BX ; un
d abspeichern.
[X14] 699: Jmp Short Copy2
[6EE] 700: Copy1: Mov Word Ptr [StepY],-1 ; Sc
hrittweite=-1
[P4G] 701: Mov AX,Word Ptr [EndY] ; Hö
he des Fensters berechnen
[XK5] 702: Mov Word Ptr [Y2],AX
[LA7] 703: Sub AX,Word Ptr [StartY]
[MCD] 704: Mov Word Ptr [DeltaY],AX ; un
d abspeichern.
[LLG] 705: Add AX,DX ; En
d-Zeile des neuen Fensters berechnen
[1SC] 706: Mov Word Ptr [Y1],AX ; un
d nach Y1 schaffen.
[LHN] 707: Copy2: Inc Word Ptr [DeltaY] ; Hö
he des Fensters war noch um 1 zu klein.
[67C] 708: Mov Byte Ptr [Richtung],1 ; Is
t XNeu>XAlt?
[WA7] 709: Cmp CX,Word Ptr [StartX]
[2A2] 710: JB Copy3
[BG7] 711: Mov Byte Ptr [Richtung],0
[EBL] 712: Copy3: Mov AX,CX ; X-
Offset der neuen Position berechnen.
[KS1] 713: Shr AX,1
[KS1] 714: Shr AX,1
[KS1] 715: Shr AX,1
[SBC] 716: Mov Word Ptr [X1],AX ; Of
fset abspeichern.
[T9M] 717: Mov AX,Word Ptr [StartX] ; X-
Offset der alten Position berechnen.
[KS1] 718: Shr AX,1
[KS1] 719: Shr AX,1
[KS1] 720: Shr AX,1
[WBC] 721: Mov Word Ptr [X2],AX ; Of
fset abspeichern.
[TYG] 722: Mov BX,Word Ptr [EndX] ; Br
eite des Fensters berechnen.

```



```

[39B] 723: ShR BX,1 ; BX
        durch 8 dividieren.
[LS1] 724: ShR BX,1
[LS1] 725: ShR BX,1
[YAF] 726: Sub BX,AX ; X-
        Offset von StartX subtrahieren.
[9F1] 727: Dec BX
[B9D] 728: Mov Word Ptr [DeltaX],BX ; We
        rt abspeichern.
[39F] 729: Mov BL,CL ; Bi
        tmuster für linken Rand holen.
[PH6] 730: And BL,7 ; (v
        gl. BitM)
[S5G] 731: Mov DL,Byte Ptr [BX+BitML]; We
        rt aus der Tabelle holen.
[MFC] 732: Mov BX,CX ; Re
        chten Rand berechnen.
[HA7] 733: Sub BX,Word Ptr [StartX]
[1F6] 734: Add BX,Word Ptr [EndX]
[4YE] 735: And BX,7 ; Bi
        tmuster für rechten Rand holen.
[UX7] 736: Mov DH,Byte Ptr [BX+BitMR]
[CEF] 737: And CL,7 ; Bi
        t-Verschiebung gegenüber altem
[R3E] 738: Mov AL,Byte Ptr [StartX] ; Fe
        nster berechnen.
[2U1] 739: And AL,7
[2B2] 740: Sub CL,AL
[BP8] 741: And CL,15 ; We
        rt steht jetzt in CL.
[268] 742: Mov Byte Ptr [Richtung+1],0
[CV1] 743: Cmp CL,8
[7A2] 744: JB Copy4
[968] 745: Mov Byte Ptr [Richtung+1],1
[U2A] 746: Neg CL ; Na
        ch links schieben!
[GMA] 747: Add CL,16 ; Ne
        uen Wert berechnen.
[T4G] 748: Pop AX ; Se
        itennummer vom Stack holen.
[LNH] 749: Mov BX,Page1 ; BX
        mit Segment der 1. Seite vorbesetzen.
[YU6] 750: Cmp AL,1 ; 1.
        Seite?
[WB2] 751: JZ Copy5
[3G8] 752: Mov BH,High Page2 ; Se
        ite 2!
[HKE] 753: Copy5: Mov DS,BX ; Se
        gment nach DS schaffen.
[2PD] 754: ; Vorbereitungen sind nun abgeschl
        ossen!
[54K] 755: Copy6: Mov AX,Word Ptr CS:[Y1] ; Ze
        ilen-Offsets berechnen.
[VF2] 756: Call Zoff
[Z1B] 757: Mov BX,AX ; Ze
        ilen-Offset nach BX.
[AB6] 758: Mov AX,Word Ptr CS:[Y2]
[VF2] 759: Call Zoff
[OZA] 760: Mov DI,AX ; Ze
        ilen-Offset nach DI.
[2F8] 761: Cmp Byte Ptr CS:[Richtung],0
[YZ5] 762: JZ CopyR ; X1
        >X2!
[DV8] 763: ; Vom linken Rand an kopieren.
[ZRH] 764: CopyL: Add BX,Word Ptr CS:[X1] ; X-
        Offset berücksichtigen.
[P76] 765: Add DI,Word Ptr CS:[X2]
[SE8] 766: Call Shift ; By
        te holen.
[4H1] 767: Not DL
[62C] 768: And Byte Ptr DS:[BX],DL ; Bi
        ts maskieren.
[E59] 769: Inc DI ; Ze
        iger erhöhen.
[4H1] 770: Not DL
[T89] 771: And AH,DL ; Bi
        ts maskieren.
[1ZB] 772: Or Byte Ptr DS:[BX],AH ; un
        d übertragen.
[G78] 773: ; Ganze Bytes übertragen.
[BG8] 774: Mov CH,Byte Ptr CS:[DeltaX]
[WVE] 775: CopyL1: Inc BX ; Zä
        hler weiterschalten.
[1Z6] 776: Or CH,CH ; Fe
        rtig?
[VS2] 777: JZ CopyL2
[SE8] 778: Call Shift ; By
        te holen.
[E59] 779: Inc DI ; Ze
        iger erhöhen.
[KFC] 780: Mov Byte Ptr DS:[BX],AH ; We
        rt abspeichern.
[LBB] 781: Dec CH ; Zä
        hler dekrementieren.

```

```

[FA4] 782: Jmp Short CopyL1
[EP7] 783: ; Rechten Rand kopieren.
[XUE] 784: CopyL2: Call Shift ; Bi
        tmuster berechnen.
[WG1] 785: Not DH
[P4B] 786: And Byte Ptr DS:[BX],DH ; Bi
        ts löschen.
[WG1] 787: Not DH
[P62] 788: And AH,DH
[D7C] 789: Or Byte Ptr DS:[BX],AH ; Bi
        ts abspeichern.
[YHA] 790: Jmp Short Copy7 ; Nä
        chste Zeile.
[MB9] 791: ; Vom rechten Rand an kopieren.
[XSH] 792: CopyR: Add BX,Word Ptr CS:[X1] ; X-
        Offset berücksichtigen.
[1F8] 793: Add BX,Word Ptr CS:[DeltaX]
[OG1] 794: Inc BX
[P76] 795: Add DI,Word Ptr CS:[X2]
[5E8] 796: Add DI,Word Ptr CS:[DeltaX]
[4F1] 797: Inc DI
[SE8] 798: Call Shift ; By
        te holen.
[WG1] 799: Not DH
[U1C] 800: And Byte Ptr DS:[BX],DH ; Bi
        ts maskieren.
[79B] 801: Dec DI ; Ze
        iger dekrementieren.
[WG1] 802: Not DH
[589] 803: And AH,DH ; Bi
        ts maskieren.
[1ZB] 804: Or Byte Ptr DS:[BX],AH ; un
        d übertragen.
[G78] 805: ; Ganze Bytes übertragen.
[BG8] 806: Mov CH,Byte Ptr CS:[DeltaX]
[NVE] 807: CopyR1: Dec BX ; Zä
        hler weiterschalten.
[1Z6] 808: Or CH,CH ; Fe
        rtig?
[TT2] 809: JZ CopyR2
[SE8] 810: Call Shift ; By
        te holen.
[79B] 811: Dec DI ; Ze
        iger dekrementieren.
[KFC] 812: Mov Byte Ptr DS:[BX],AH ; We
        rt abspeichern.
[LBB] 813: Dec CH ; Zä
        hler dekrementieren.
[1B4] 814: Jmp Short CopyR1
[PD7] 815: ; Linken Rand kopieren.
[VVE] 816: CopyR2: Call Shift ; Bi
        tmuster berechnen.
[4H1] 817: Not DL
[25B] 818: And Byte Ptr DS:[BX],DL ; Bi
        ts löschen.
[4H1] 819: Not DL
[A72] 820: And AH,DL
[D7C] 821: Or Byte Ptr DS:[BX],AH ; Bi
        ts abspeichern.
[1AH] 822: Copy7: Dec Word Ptr CS:[DeltaY] ; Ze
        ilenzahl verringern.
[FC7] 823: JZ CopyE ; Fe
        rtig?
[GRE] 824: Mov AX,Word Ptr CS:[Y1] ; Y-
        Werte aktualisieren.
[LU7] 825: Add AX,Word Ptr CS:[StepY]
[546] 826: Mov Word Ptr CS:[Y1],AX
[AB6] 827: Mov AX,Word Ptr CS:[Y2]
[LU7] 828: Add AX,Word Ptr CS:[StepY]
[C46] 829: Mov Word Ptr CS:[Y2],AX
[9R9] 830: Jmp Copy6 ; Nä
        chste Zeile.
[VW8] 831: CopyE: Ret ; Fe
        rtig!
[000] 832:
[M6M] 833: ; Rechteckige Fläche mit Bitmuster
        ausfüllen (X1=DX, Y1=DX, X2=DI, Y2=SI).
[9K8] 834: FilPt: Cmp CX,DI ; X1
        >X2?
[FV2] 835: JNA FilPt1
[MHA] 836: Xchg CX,DI ; X-
        Werte tauschen.
[XS8] 837: FilPt1: Cmp DX,SI ; Y1
        >Y2?
[NV2] 838: JNA FilPt2
[KLA] 839: Xchg DX,SI ; Y-
        Werte tauschen.
[HRK] 840: FilPt2: Add CX,Word Ptr [StartX] ; Fe
        nster berücksichtigen.
[897] 841: Add DX,Word Ptr [StartY]
[787] 842: Add DI,Word Ptr [StartX]
[T87] 843: Add SI,Word Ptr [StartY]
[PGL] 844: Cmp CX,Word Ptr [StartX] ; Pr
        üfen, ob Koordinaten innerhalb des

```



```

[CLD] 845: JAE FilPt3 ; ak
[WB7] 846: Mov CX,Word Ptr [StartX]
[5BH] 847: FilPt3: Cmp CX,Word Ptr [EndX] ; X1
[NB9] 848: JA CopyE ; We
nn ja --> Ende.
[XRF] 849: Cmp DI,Word Ptr [StartX] ; X2
- Koordinate zu klein?
[RB9] 850: JB CopyE ; We
nn ja --> Ende.
[K2F] 851: Cmp DI,Word Ptr [EndX] ; X2
- Koordinate zu groß?
[NV2] 852: JBE FilPt4
[1CG] 853: Mov DI,Word Ptr [EndX] ; X2
- Koordinate = rechter Rand.
[RMH] 854: FilPt4: Cmp DX,Word Ptr [StartY] ; Y1
- Koordinate zu klein?
[TV2] 855: JAE FilPt5
[U1G] 856: Mov DX,Word Ptr [StartY] ; Y1
- Koordinate = oberer Rand.
[RBH] 857: FilPt5: Cmp DX,Word Ptr [EndY] ; Y1
- Koordinate zu groß?
[NB9] 858: JA CopyE ; We
nn ja --> Ende.
[Gsf] 859: Cmp SI,Word Ptr [StartY] ; Y2
- Koordinate zu klein?
[RB9] 860: JB CopyE ; We
nn ja --> Ende.
[33F] 861: Cmp SI,Word Ptr [EndY] ; Y2
- Koordinate zu groß?
[1W2] 862: JBE FilPt6
[4EG] 863: Mov SI,Word Ptr [EndY] ; Y2
- Koordinate = unterer Rand.
[HDH] 864: FilPt6: Mov BX,DI ; Bi
tmuster für rechten Rand holen
[5W6] 865: And BX,7 ; (v
gl BitM).
[RX7] 866: Mov AH,Byte Ptr [BX+BitMR]
[39F] 867: Mov BL,CL ; Bi
tmuster für linken Rand holen.
[3U1] 868: And BL,7
[128] 869: Mov AL,Byte Ptr [BX+BitM1]
[W4E] 870: Shr CX,1 ; By
te-Offset der beiden Ränder be-
[W47] 871: Shr CX,1 ; re
chnen.
[MS1] 872: Shr CX,1
[RR1] 873: Shr DI,1
[RR1] 874: Shr DI,1
[RR1] 875: Shr DI,1
[60F] 876: Sub DI,CX ; Di
fferenz der Offsets berechnen.
[8EE] 877: JZ FilPtS ; Sc
hmales Rechteck ausfüllen.
[8SC] 878: FilPt7: Push AX ; Bi
tmasken retten.
[EEA] 879: Mov AX,DX ; Y-
Koordinate holen.
[BTC] 880: Call ZOff ; Ze
ilen-Offset berechnen.
[SKC] 881: Mov BX,AX ; Ad
resse nach BX schaffen.
[P2D] 882: Add BX,CX ; X1
-Offset berücksichtigen.
[NCC] 883: Pop AX ; Bi
tmuster vom Stack holen.
[TH1] 884: Not AX
[H56] 885: And Byte Ptr ES:[BX],AL
[872] 886: Add BX,DI
[556] 887: And Byte Ptr ES:[BX],AH
[TH1] 888: Not AX
[VTD] 889: Push BX ; Ad
resse auf dem Stack retten.
[10D] 890: Mov BX,DX ; Bi
tmuster zum Füllen holen.
[VU1] 891: And BX,7
[K19] 892: Mov CH,Byte Ptr [BX+Pattern]
[8FC] 893: Pop BX ; Ad
resse vom Stack holen.
[E8F] 894: Push CX ; Fü
llmuster auf dem Stack retten.
[C62] 895: And CH,AH
[WNE] 896: Or Byte Ptr ES:[BX],CH ; Re
chten Rand schreiben.
[5TD] 897: Pop CX ; Fü
llmuster wiederherstellen.
[SS8] 898: Push DI ; Br
eite retten.
[G2F] 899: FilPt8: Dec BX ; Ad
resse dekrementieren.
[1LA] 900: Dec DI ; Li
nker Rand erreicht?
[GY9] 901: JZ FilPt9 ; We
nn ja --> FilPt9.

```

```

[D76] 902: Mov Byte Ptr ES:[BX],CH
[4RA] 903: Jmp Short FilPt8 ; Nä
chstes Byte.
[8ME] 904: FilPt9: And CH,AL ; Li
nken Rand schreiben.
[L25] 905: Or Byte Ptr ES:[BX],CH
[2CC] 906: Pop DI ; Br
eite wiederherstellen.
[2RA] 907: Mov CH,0 ; CX
wiederherstellen.
[ECD] 908: Inc DX ; Ze
ilennummer weiterschalten.
[U2C] 909: Cmp DX,SI ; Un
terer Rand erreicht?
[7W2] 910: JBE FilPt7
[9G6] 911: Ret ; Fe
rtig!
[FTA] 912: ; Sehr schmale Rechtecke ausfüllen
[H8G] 913: FilPtS: And AL,AH ; Ne
ue Bitmaske in AL aufsetzen.
[SOD] 914: FilPtS1: Mov BX,DX ; Fü
llmuster holen.
[VU1] 915: And BX,7
[G19] 916: Mov AH,Byte Ptr [BX+Pattern]
[NCA] 917: Push AX ; Bi
tmuster retten.
[UGC] 918: Mov AX,DX ; Ze
ilen-Offset berechnen.
[VF2] 919: Call ZOff
[SKC] 920: Mov BX,AX ; Ad
resse nach BX schaffen.
[CYD] 921: Add BX,CX ; X-
Koordinate berücksichtigen.
[NCC] 922: Pop AX ; Bi
tmuster vom Stack holen.
[1H1] 923: Not AL
[Y4B] 924: And Byte Ptr ES:[BX],AL ; Bi
ts löschen.
[1H1] 925: Not AL
[61F] 926: And AH,AL ; Bi
ts des Füllmusters maskieren.
[TYC] 927: Or Byte Ptr ES:[BX],AH ; Fü
llmuster ablegen.
[MY8] 928: Inc DX ; Nä
chste Zeile.
[U2C] 929: Cmp DX,SI ; Un
terer Rand erreicht?
[6C3] 930: JBE FilPtS1
[9G6] 931: Ret ; Fe
rtig!
[000] 932:
[NAD] 933: Include Hercu12.ASM ; Re
st des Programms.

```

Listing 1. Der erste Teil des Assembler-Quellcodes für »hercules.com«

```

[LM8] 0: ; Ende des Programms GSYS1.ASM.
[000] 1:
[H8M] 2: ; Füllmuster für FilPt-Befehl defi
nieren. Das Muster wird in CX, DX,
[5D6] 3: ; DI und SI übergeben.
[XCK] 4: DefPt: Mov Word Ptr [Pattern],CX ; Bi
tmuster abspeichern.
[7E8] 5: Mov Word Ptr [Pattern+2],DX
[7D8] 6: Mov Word Ptr DS:[Pattern+4],DI
[4E8] 7: Mov Word Ptr [Pattern+6],SI
[SLO] 8: Ret
[000] 9:
[21C] 10: ; Kreis zeichnen (XO=CX, YO=DX, Ra
dius=DI).
[YYG] 11: Circle: Mov BX,DI ; Ra
dius nach BX und SI schaffen.
[PB2] 12: Mov SI,BX
[WGB] 13: Shr BX,1 ; Ra
dius durch 4 dividieren.
[LS1] 14: Shr BX,1
[06E] 15: Sub SI,BX ; Y-
Halbmesser berechnen (YH=0.75*XH).
[000] 16:
[7GB] 17: ; Ellipse zeichnen (XO=CX, YO=DX,
A=DI, B=SI).

```



```

[OPK] 18: Ellip: Add CX,Word Ptr [StartX] ; Fenster berücksichtigen.
[897] 19: Add DX,Word Ptr [StartY]
[UHK] 20: Mov Word Ptr [X1],CX ; Koordinaten des Mittelpunkts merken.
[EL5] 21: Mov Word Ptr [Y1],DX
[2W6] 22: Mov Word Ptr [X2],0 ; X=0
[H57] 23: Mov Word Ptr [Y2],SI ; Y=B
[BE7] 24: Push DI ; retten.
[NSG] 25: Dec DI ; dius war für Berechnungen um 1 zu groß.
[5V8] 26: Mov AX,SI ; B berechnen.
[OH1] 27: Mul SI
[EEG] 28: Mov Word Ptr [DeX],AX ; rechneten Wert abspeichern.
[Y06] 29: Mov Word Ptr [DeX+2],DX
[89C] 30: Mov Word Ptr [DeX+4],0 ; f 48 Bit erweitern.
[8KC] 31: ShL AX,1 ; rt mit 2 multiplizieren
[SN1] 32: RCL DX,1
[H1F] 33: Mov Word Ptr [QB],AX ; d nach QB schaffen --> QB=2*B*B.
[7V5] 34: Mov Word Ptr [QB+2],DX
[5T8] 35: Mov AX,DI ; A berechnen.
[HG1] 36: Mul DI
[M9D] 37: Mov Word Ptr [DeA],AX ; rt nach DA schaffen.
[BX5] 38: Mov Word Ptr [DeA+2],DX
[RY7] 39: Push AX ; rt retten.
[7V1] 40: Push DX
[8KC] 41: ShL AX,1 ; rt mit 2 multiplizieren
[SN1] 42: RCL DX,1
[9UC] 43: Mov Word Ptr [QA],AX ; d nach QA schaffen.
[3V5] 44: Mov Word Ptr [QA+2],DX
[7DG] 45: Pop AX ; rt wieder vom Stack holen (High-Word).
[769] 46: Mov BX,SI ; B-1 berechnen.
[1S1] 47: ShL BX,1
[9F1] 48: Dec BX
[E1E] 49: Mul BX ; *B-1)*A*A berechnen (=2*A*A*B-A*A=QA*B-A*A).
[89C] 50: Mov Word Ptr [DeY+2],AX ; rt abspeichern.
[H16] 51: Mov Word Ptr [DeY+4],DX
[SR5] 52: Pop AX ; w-Word berücksichtigen.
[DH1] 53: Mul BX
[LRD] 54: Mov Word Ptr [DeY],AX ; w-Word abspeichern.
[UBC] 55: Add Word Ptr [DeY+2],DX ; gh-Word addieren.
[DAF] 56: AdC Word Ptr [DeY+4],0 ; ertrag berücksichtigen.
[MXB] 57: Mov AX,Word Ptr [DeX] ; A*B*B berechnen.
[HR5] 58: Add Word Ptr [DeA],AX
[P1D] 59: Mov AX,Word Ptr [DeX+2] ; gh-Word addieren.
[AR5] 60: AdC Word Ptr [DeA+2],AX
[N5C] 61: Mov Word Ptr [DeA+4],0 ; f 48 Bit erweitern.
[DB9] 62: Pop SI ; vom Stack holen.
[Y2F] 63: Ellip1: Mov CX,Word Ptr [X1] ; Koordinaten holen
[PZ5] 64: Mov DX,Word Ptr [Y1]
[ML7] 65: Add CX,Word Ptr [X2] ; +X
[OM7] 66: Add DX,Word Ptr [Y2] ; +Y
[TPA] 67: Call Point1 ; nkt zeichnen.
[YTC] 68: Mov CX,Word Ptr [X1] ; ordinatoraten holen.
[PZ5] 69: Mov DX,Word Ptr [Y1]
[8N7] 70: Sub CX,Word Ptr [X2] ; -X
[OM7] 71: Add DX,Word Ptr [Y2] ; +Y
[TPA] 72: Call Point1 ; nkt zeichnen.
[YTC] 73: Mov CX,Word Ptr [X1] ; ordinatoraten holen.

```

```

[PZ5] 74: Mov DX,Word Ptr [Y1]
[ML7] 75: Add CX,Word Ptr [X2] ; +X
[MN7] 76: Sub DX,Word Ptr [Y2] ; -Y
[TPA] 77: Call Point1 ; nkt zeichnen.
[YTC] 78: Mov CX,Word Ptr [X1] ; ordinatoraten holen.
[PZ5] 79: Mov DX,Word Ptr [Y1]
[8N7] 80: Sub CX,Word Ptr [X2] ; -X
[MN7] 81: Sub DX,Word Ptr [Y2] ; -Y
[TPA] 82: Call Point1 ; nkt zeichnen.
[CX7] 83: Test Byte Ptr [DeA+5],128 ; >=0
[HY5] 84: JnZ Ellip2 ; <0!
[457] 85: Inc Word Ptr [X2] ; X+1
[R09] 86: Mov AX,Word Ptr [DeX] ; =DA-DX
[ZS5] 87: Sub Word Ptr [DeA],AX
[7M6] 88: Mov AX,Word Ptr [DeX+2]
[NR5] 89: SBB Word Ptr [DeA+2],AX
[BM6] 90: Mov AX,Word Ptr [DeX+4]
[3S5] 91: SBB Word Ptr [DeA+4],AX
[319] 92: Mov AX,Word Ptr [QB] ; =DX+QB
[4V5] 93: Add Word Ptr [DeX],AX
[696] 94: Mov AX,Word Ptr [QB+2]
[XU5] 95: AdC Word Ptr [DeX+2],AX
[8AF] 96: AdC Word Ptr [DeX+4],0 ; ertrag berücksichtigen.
[DD4] 97: Jump Short Ellip3
[4S9] 98: Ellip2: Dec Word Ptr [Y2] ; Y-1
[B09] 99: Mov AX,Word Ptr [DeY] ; =DA+DY
[HR5] 100: Add Word Ptr [DeA],AX
[7M6] 101: Mov AX,Word Ptr [DeY+2]
[AR5] 102: AdC Word Ptr [DeA+2],AX
[BM6] 103: Mov AX,Word Ptr [DeY+4]
[SR5] 104: AdC Word Ptr [DeA+4],AX
[F19] 105: Mov AX,Word Ptr [QA] ; =DY-QA
[RW5] 106: Sub Word Ptr [DeY],AX
[Z86] 107: Mov AX,Word Ptr [QA+2]
[DV5] 108: SBB Word Ptr [DeY+2],AX
[SAF] 109: SBB Word Ptr [DeY+4],0 ; ertrag berücksichtigen.
[4H9] 110: Ellip3: Cmp Word Ptr [Y2],0 ; =0
[322] 111: JS EllipE
[SN5] 112: Cmp Word Ptr [X2],SI
[ZX2] 113: JA EllipE
[TSA] 114: Jump Ellip1 ; chste Punkte.
[C93] 115: EllipE: Ret
[000] 116: ; Bitmuster zum Füllen bis zur 1.
[3WH] 117: ; Byte-Grenze berechnen.
[4FM] 118: BitM: Mov BX,Word Ptr [StartX] ; nken Rand des Fensters holen.
[UFA] 119: And BX,7 ; ere Bits entfernen.
[42G] 120: Mov CL,Byte Ptr [BX+BitML] ; rt aus Tabelle holen.
[HCK] 121: Mov BL,Byte Ptr [EndX] ; chten Rand des Fensters holen.
[2FA] 122: And BL,7 ; ere Bits entfernen.
[U2G] 123: Mov CH,Byte Ptr [BX+BitMR] ; rt aus Tabelle holen.
[4L6] 124: Ret ; rtig.
[000] 125: ; Byte holen und verschieben.
[RW8] 126: ;
[S3E] 127: Shift: Mov AH,Byte Ptr ES:[DI] ; 1. Byte holen.
[5B8] 128: Cmp Byte Ptr CS:[Richtung+1],0
[WAB] 129: JnZ ShiftL ; ch links schieben.
[ERB] 130: Mov AL,Byte Ptr ES:[DI-1] ; 2. Byte holen.
[TZ9] 131: RoR AX,CL ; rt verschieben.
[9G6] 132: Ret ; rtig!
[3NE] 133: ShiftL: Mov AL,Byte Ptr ES:[DI+1] ; 2. Byte holen.
[RFA] 134: RoL AX,CL ; ch links schieben.

```



```
[9G6] 135: Ret ; Fe
[000] 136: rtig!
[2DK] 137: ; Zeichen an die Stelle X,Y schrei
[DKC] 138: DrawChr: Test AL,128 ; Ze
ichencode>128?
[2T3] 139: JZ DrawChr1
[69F] 140: XOr BX,BX ; Ve
ktor auf Zeichentabelle holen.
[VB2] 141: Mov DS,BX
[PX8] 142: Mov SI,Word Ptr DS:[GrafTabl*4]
]
[YD9] 143: Mov DS,Word Ptr DS:[GrafTabl*4]
+2]
[KDB] 144: And AL,127 ; Ob
erstes Bit löschen.
[AB5] 145: Jmp Short DrawChr2
[4HL] 146: DrawChr1: Mov BX,BIOSeg ; Ze
ichengenerator im BIOS verwenden.
[VB2] 147: Mov DS,BX
[F34] 148: Mov SI,CharGen
[D4L] 149: DrawChr2: Mov AH,0 ; Ze
ichencode auf 16 bit erweitern und
[SHA] 150: ShL AX,1 ; mi
t 8 multiplizieren.
[OS1] 151: ShL AX,1
[OS1] 152: ShL AX,1
[UCG] 153: Add SI,AX ; Ad
resse des 1. Bytes steht jetzt in SI.
[X4C] 154: Mov DI,CX ; X-
Koordinate nach DI und
[8N9] 155: ShR DI,1 ; du
rch 8 dividieren.
[RR1] 156: ShR DI,1
[RR1] 157: ShR DI,1
[G9G] 158: And CL,7 ; Za
hl der Verschiebungsschritte nach CL.
[5A9] 159: Mov CH,8 ; Zä
hler für 8 Bytes.
[T4E] 160: DrawChr3: Mov AX,DX ; Y-
Koordinate nach AX.
[BTC] 161: Call Zoff ; Ze
ilen-Offset berechnen.
[R3A] 162: Push DI ; X-
Offset retten.
[AKC] 163: Add DI,AX ; Ko
mplette Adresse nach DI.
[CDP] 164: LodSB ; By
te aus Zeichengenerator holen.
[GD9] 165: Mov AH,AL ; na
ch AH schaffen
[WWB] 166: Mov AL,0 ; un
d auf 16 Bit erweitern.
[FTD] 167: ShR AX,CL ; Bi
t-Position berücksichtigen.
[7G2] 168: Xchg AH,AL
[E0E] 169: Mov BH,Byte Ptr CS:[Mode] ; Ze
ichen-Modus holen.
[B8A] 170: Or BH,BH ; Ze
ichen schreiben?
[S23] 171: JnZ DrawChr4
[9NC] 172: Or Word Ptr ES:[DI],AX ; Ze
ichen schreiben.
[2C5] 173: Jmp Short DrawChr6
[7NC] 174: DrawChr4: Dec BH ; Ze
ichen löschen?
[S23] 175: JnZ DrawChr5
[TH1] 176: Not AX
[OTC] 177: And Word Ptr ES:[DI],AX ; Ze
ichen löschen.
[2C5] 178: Jmp Short DrawChr6
[LLG] 179: DrawChr5: XOr Word Ptr ES:[DI],AX ; Ze
ichen invertieren.
[ESF] 180: DrawChr6: Pop DI ; X-
Offset wiederherstellen.
[MY8] 181: Inc DX ; Nä
chste Zeile.
[DS6] 182: Dec CH ; Fe
rtig?
[S23] 183: JnZ DrawChr3
[HXA] 184: Push CS ; DS
wiederherstellen.
[8H1] 185: Pop DS
[SLO] 186: Ret
[000] 187: ;
[758] 188: ; Timer-Interrupt-Routine.
[95A] 189: TimInt: Push DS ; DS
retten.
[RCA] 190: Push CS ; CS
nach DS schaffen.
[8H1] 191: Pop DS
[C88] 192: Cmp Byte Ptr [Graphic],Off
[6SB] 193: JZ TimIntE ; Cu
rsor angeschaltet?
```

```
[XW7] 194: Cmp Byte Ptr [Cursor],Off
[74C] 195: JZ TimIntE ; Gr
afik eingeschaltet?
[42D] 196: StI ; In
terrupts wieder zulassen.
[MLL] 197: Dec Byte Ptr [Delay] ; Ve
rzögerungszähler dekrementieren.
[6U3] 198: JnZ TimIntE
[4K8] 199: Mov Byte Ptr [Delay],Blink
[SCA] 200: Push BX ; Re
gister retten.
[YU1] 201: Push ES
[5AB] 202: Mov BX,Word Ptr [Page] ; Se
ite setzen.
[WB2] 203: Mov ES,BX
[MPB] 204: Call DrawCur ; Cu
rsor zeichnen.
[L5D] 205: Pop ES ; Re
gister wiederherstellen.
[GH1] 206: Pop BX
[BND] 207: TimIntE: Pop DS ; DS
wiederherstellen.
[EOK] 208: Jmp DWord Ptr CS:[OldInt] ; Al
te Interrupt-Routine aufrufen.
[000] 209: ;
[C9R] 210: ; Zeichen an die Stelle X,Y schrei
ben (CX=X, DX=Y, XZoom=BL, YZoom=BH, Zeichen
in AL).
[64E] 211: BigChr: Push BX ; Ve
rgrößerung retten.
[4DL] 212: Mov Word Ptr CS:[X1],CX ; X-
und Y-Koordinate zwischenspeichern.
[B46] 213: Mov Word Ptr CS:[Y1],DX
[9F9] 214: Test AL,128 ; Ze
ichencode>128?
[YB3] 215: JZ BigChr1
[69F] 216: XOr BX,BX ; Ve
ktor auf Zeichentabelle holen.
[VB2] 217: Mov DS,BX
[PX8] 218: Mov SI,Word Ptr DS:[GrafTabl*4]
]
[YD9] 219: Mov DS,Word Ptr DS:[GrafTabl*4]
+2]
[KDB] 220: And AL,127 ; Ob
erstes Bit löschen.
[5T4] 221: Jmp Short BigChr2
[63L] 222: BigChr1: Mov BX,BIOSeg ; Ze
ichengenerator im BIOS verwenden.
[VB2] 223: Mov DS,BX
[F34] 224: Mov SI,CharGen
[18K] 225: BigChr2: Mov AH,0 ; Ze
ichencode auf 16 bit erweitern und
[SHA] 226: ShL AX,1 ; mi
t 8 multiplizieren.
[OS1] 227: ShL AX,1
[OS1] 228: ShL AX,1
[UCG] 229: Add SI,AX ; Ad
resse des 1. Bytes steht jetzt in SI.
[OPF] 230: Pop BX ; Ve
rgrößerung wiederherstellen.
[Z9H] 231: Mov DX,1 ; Bi
tmuster für einen Punkt in X-Richtung.
[MSB] 232: Mov CL,BL ; X-
Vergrößerung nach CL.
[NW9] 233: Mov CH,0 ; Au
f 16 Bit erweitern.
[ABD] 234: BigChr3: Or DH,DL ; Bi
tmuster erzeugen.
[BR1] 235: ShL DL,1
[523] 236: Loop BigChr3
[CL7] 237: Mov DL,0 ; DL
löschen.
[34C] 238: Mov Byte Ptr CS:[Y2],8 ; Zä
hler für 8 Zeilen.
[AK8] 239: Mov Byte Ptr CS:[Richtung],BH
[HLF] 240: BigChr4: Mov AX,Word Ptr CS:[Y1] ; Y-
Koordinate holen.
[X3D] 241: Push Word Ptr CS:[X1] ; X-
Koordinate retten.
[4TF] 242: Call Zoff ; Ze
ilenoffset berechnen und retten.
[4V1] 243: Push AX
[AZH] 244: Mov AL,Byte Ptr DS:[SI] ; Bi
tmuster des Buchstabens holen.
[9MC] 245: Mov Byte Ptr CS:[X2],8 ; Zä
hler für 8 Spalten.
[82C] 246: BigChr5: RoL AL,1 ; Pu
nkt abprüfen.
[P5B] 247: JnC BigChr9 ; Pu
nkt nicht setzen!
[96K] 248: Mov CX,Word Ptr CS:[X1] ; X-
Koordinate holen und Verschiebung
[2C7] 249: And CL,7 ; be
rechnen.
```



```

[LPE] 250: Pop DI ; Ze
[AT8] 251: Push DI ; wi
[SGA] 252: Push DX ; Bi
[Y7C] 253: Ror DX,CL ; Bi
[PG2] 254: Xchg DH,DL
[K1D] 255: Mov CL,Byte Ptr CS:[X1] ; X-
Offset berechnen.

[2HE] 256: Shr CX,1 ; X-
Koordinate durch 8 dividieren.
[MS1] 257: Shr CX,1
[MS1] 258: Shr CX,1
[S7E] 259: Add DI,CX ; X-
Offset zum Y-Offset addieren.
[LCG] 260: Mov AH,Byte Ptr CS:[Mode] ; Ze
[68A] 261: Or AH,AH ; Ze
[RL3] 262: Jnz BigChr6
[XNC] 263: Or Word Ptr ES:[DI],DX ; Bi
tmuster odern.
[3U4] 264: Jmp Short BigChr8
[T7C] 265: BigChr6: Dec AH ; Ze
[YL3] 266: Jnz BigChr7
[SGF] 267: Not DX ; Bi
tmuster invertieren und unden.
[N36] 268: And Word Ptr ES:[DI],DX
[3U4] 269: Jmp Short BigChr8
[S7G] 270: BigChr7: Xor Word Ptr ES:[DI],DX ; Ze
[SA4G] 271: BigChr8: Pop DX ; Bi
tmuster wiederherstellen.
[6PK] 272: BigChr9: Add Byte Ptr CS:[X1],BL ; X-
Koordinate weiterschalten.
[DLF] 273: AdC Byte Ptr CS:[X1+1],0 ; Ub
[34C] 274: Dec Byte Ptr CS:[X2] ; Na
[GL3] 275: Jnz BigChr5
[Z2C] 276: Pop AX ; Y-
Offset vom Stack holen.
[9KE] 277: Pop Word Ptr CS:[X1] ; X-
Offset vom Stack holen.
[37C] 278: Inc Word Ptr CS:[Y1] ; Na
[LSZH] 279: Dec Byte Ptr CS:[Richtung]; Ze
[A8D] 280: Jnz BigChr4 ; Ze
[AK8] 281: Mov Byte Ptr CS:[Richtung],BH
[XRC] 282: Inc SI ; Na
[YFE] 283: Dec Byte Ptr CS:[Y2] ; Al
[168] 284: JZ DrawCurE ; Fe
[9N3] 285: Jmp BigChr4
[000] 286:
[ZS5] 287: ; Cursor zeichnen.
[KAD] 288: DrawCur: Push AX ; Re
gister retten.
[6V1] 289: Push CX
[7V1] 290: Push DX
[BZD] 291: Mov BX,Word Ptr [StartX] ; X-
Offset berechnen.
[KC9] 292: Mov CL,BL ; Lo
w-Byte nach CL.
[LS1] 293: Shr BX,1
[LS1] 294: Shr BX,1
[LS1] 295: Shr BX,1
[TCG] 296: Add BX,Word Ptr [XPos] ; X-
Position berücksichtigen.
[VGD] 297: And CL,7 ; Za
hl der Schiebe-Operationen.
[H6B] 298: Mov DX,OFF00H ; Bi
tmaske erstellen.
[T62] 299: Shr DX,CL
[PG2] 300: Xchg DH,DL
[G7D] 301: Mov CX,Word Ptr [Hoehe] ; Hö
he des Cursors.
[50D] 302: Mov AX,Word Ptr [YPos] ; Y-
Position holen.
[SHA] 303: ShL AX,1 ; mi
t 8 multiplizieren.
[OS1] 304: ShL AX,1
[OS1] 305: ShL AX,1
[UXG] 306: Add AX,Word Ptr [StartY] ; Fe
nster berücksichtigen.
[XUG] 307: Add AX,Word Ptr [StartZ] ; St
art-Zeile berücksichtigen.

```

```

[CBD] 308: DrawCur1: Push AX ; Re
gister retten.
[5V1] 309: Push BX
[BTC] 310: Call ZOff ; Ze
[CS3D] 311: Add BX,AX ; X-
Offset berücksichtigen.
[XWC] 312: Xor Word Ptr ES:[BX],DX ; Pu
nkte invertieren.
[U5D] 313: Pop BX ; Re
gister wiederherstellen.
[FH1] 314: Pop AX
[50B] 315: Inc AX ; Ze
[316] 316: Loop DrawCur1 ; Nä
chste Zeile.
[21E] 317: Xor Byte Ptr [CursMod],On ; Zu
stand wechseln.
[W5D] 318: Pop DX ; Re
gister wiederherstellen.
[HH1] 319: Pop CX
[FH1] 320: Pop AX
[FT3] 321: DrawCurE: Ret
[000] 322:
[MG8] 323: ; Zeilen-Offset berechnen.
[Y99] 324: Zoff: Push BX ; BX
retten.
[WHF] 325: Mov BX,AX ; De
r Offset berechnet sich wie folgt:
[6U7] 326: Shr AX,1 ; Of
s = 90 * (Y Div 4)
[VF5] 327: Shr AX,1 ;
+
[H78] 328: Mov AH,90 ;
2000H * (Y Mod 4)
[CG1] 329: Mul AH
[BU1] 330: And BX,3
[OT1] 331: Ror BX,1
[OT1] 332: Ror BX,1
[OT1] 333: Ror BX,1
[A92] 334: Add AX,BX
[GH1] 335: Pop BX
[4L6] 336: Ret ; Fe
rtig.
[000] 337:
[OE3] 338: ; Variablen:
[VD7] 339: ; Allgemeine Variablen:
[5FF] 340: Graphic: DB Off ; Gr
afik an-/ausgeschaltet.
[3EF] 341: Page: DW Page1 ; Se
ite, die beschrieben wird.
[4BH] 342: Mode: DB 0 ; Ze
ichen-Modus (0=schreiben, 1=löschen,
[HE5] 343: invertieren). ; 2=
[7HF] 344: LinStyle:DW OFFFFH ; Au
ssuchen von einer Linie.
[1UG] 345: OldInt: DD 0 ; Pl
atz für alten Timer-Interrupt.
[WH7] 346: ; Variable für Fenster:
[6DE] 347: StartX: DW 0 ; X-
Startwert des Fensters.
[22E] 348: EndX: DW XMax ; X-
Endwert des Fensters.
[EDE] 349: StartY: DW 0 ; Y-
Startwert des Fensters.
[A2E] 350: EndY: DW YMax ; Y-
Endwert des Fensters.
[BN5] 351: ; Cursor-Variable:
[6SD] 352: XPos: DW 0 ; X-
Position des Cursors.
[9SD] 353: YPos: DW 0 ; Y-
Position des Cursors.
[BUN] 354: StartZ: DW 0 ; Sc
an-Line, ab der der Cursor ein Zeichen inver
tiert.
[BTF] 355: Hoehe: DW 8 ; De
r Cursor ist 8 Scan-Lines hoch.
[SUH] 356: Delay: DB 1 ; Ve
rögerungszähler für Blinken.
[7RF] 357: CursMod: DB Off ; Cu
rsor (nicht-)gezeichnet.
[V9F] 358: Cursor: DB On ; Cu
rsor an-/ausgeschaltet.
[FRL] 359: CursSave:DB On ; Zw
ischenspeicher für Variable 'Cursor'.
[000] 360:
[47B] 361: ; Füllmuster für FillPattern-Befeh
l:
[5NG] 362: Pattern: DB 8 Dup(?) ; Wi
rd erst später definiert.
[000] 363:
[613] 364: ; Tabellen:
[U3F] 365: ; Tabelle mit Einsprungsadressen de

```



```

r Kommandos:
[GP] 366: CmdTab: DW SetMode,GetMode,SetCurs,Se
tPos,GetPos,SetMod,Disp,SetPage
[MAK] 367: DW Clear,ScrCopy,Swap,Char,Pr
int,PrintB,ScrUp,Point,GetCol
[XZK] 368: DW SetStyle,Line,Circle,Ellip
,FillShp,FilPt,DefPt,DefWin,FilWin
[DB4] 369: DW Invert,Copy
[5DE] 370: ; Bitmuster für Füllen bis zur 1.
Byte-Grenze.
[5DF] 371: BitML: DB 255,127,63,31,15,7,3,1; We
rte für linke Byte-Grenze.
[6XH] 372: BitMR: DB 128,192,224,240,248,252,25
4,255; Werte für recht Byte-Grenze.
[AGA] 373: ; Werte um auf Grafik zu schalten:
[M7C] 374: GTable: DB 35H,2DH,2EH,07H,5BH,02H,57
H,57H,02H,03H,00H,00H
[43A] 375: ; Werte um auf Text zu schalten:
[PEC] 376: TTable: DB 61H,50H,52H,0FH,19H,06H,19
H,19H,02H,0DH,0BH,0CH
[000] 377:
[2F8] 378: ; Programm installieren.
[WEH] 379: Init: Mov AX,3500H+Timer ; Al
ten Timer-Interrupt merken.
[WS1] 380: Int DOS
[5P6] 381: Mov Word Ptr OldInt,BX
[PF7] 382: Mov Word Ptr OldInt[2],ES
[VMG] 383: Mov SI,Offset TimInt ; Pr
üfen, ob das Programm schon
[3DA] 384: Mov DI,BX ; in
stalliert ist.
[H46] 385: Mov CX,TimIntE-TimInt
[3P2] 386: RepE CmpSB
[DSH] 387: JCKZ Onlyl ; Da
s Programm ist bereits installiert!
[Y1K] 388: Mov DX,Offset TimInt ; Ve
ktor auf neue Routine biegen.
[AV4] 389: Mov AX,2500H+Timer
[WS1] 390: Int DOS
[ZRK] 391: Mov DX,Offset NInt ; In
terruptvektor auf
[C69] 392: Mov AL,VideoInt ; ve
rbiegen.
[WS1] 393: Int DOS
[FOF] 394: Mov DX,Offset Meldung ; St
artmeldung ausgeben.
[6W1] 395: Mov AH,9
[WS1] 396: Int DOS
[OE] 397: Mov DX,Offset Ende ; Pr
ogramm resident machen.
[E54] 398: Int Resident
[P4H] 399: Onlyl: Mov DX,Offset NoMore ; Fe
hlermeldung ausgeben.
[6W1] 400: Mov AH,9
[WS1] 401: Int DOS
[B26] 402: Int Quit ; En
de.
[000] 403:
[T4P] 404: Meldung: DB 'Grafik-Treiber für Hercul
es-Karte V 1.1 (C) F. Brendle 1987.',13,10,
'$'
[KTH] 405: NoMore: DB 'Der Grafik-Treiber ist be
reits installiert!'
[082] 406: DB 10,13,'$'
[000] 407:
[CFG] 408: ; Variablen, die nur vorübergehend
benötigt werden:
[9X7] 409: ; Variablen für Linien:
[EK9] 410: X1: DW 0 ; X1
-Koordinate.
[HK9] 411: Y1: DW 0 ; Y1
-Koordinate.
[LK9] 412: X2: DW 0 ; X2
-Koordinate.
[PK9] 413: Y2: DW 0 ; Y2
-Koordinate.
[1RE] 414: DeltaX: DW 0 ; Zw
ischenspeicher für X2-X1.
[GTF] 415: DeltaY: DW 0 ; Zw
ischenspeicher für Abs(Y2-Y1).
[FHC] 416: Richtung: DW 0 ; Ze
ichen-Richtung.
[NCE] 417: StepY: DW 0 ; Sc
hrittweite in Y-Richtung.
[H9D] 418: ; Variablen für Ellipsen (48-Bit-R
egister).
[MFA] 419: DeX: DB 6 Dup(?) ; Vg
l. c't 10/86, S.12.
[D94] 420: DeY: DB 6 Dup(?)
[574] 421: DeA: DB 6 Dup(?)
[WXA] 422: QA: DB 4 Dup(?) ; 32
-Bit-Register.
[524] 423: QB: DB 4 Dup(?)
[000] 424:
[AD9] 425: ; 1. Puffer für Flächenfüllen.
[DAF] 426: Buffer1: DB (YMax*8+9+Offset Init-Offs
et Buffer1) Dup(?)

```

```

[DD9] 427: ; 2. Puffer für Flächenfüllen.
[KH7] 428: Buffer2: DB (YMax*8+9 Dup(?))
[381] 429: Ende:
[000] 430:
[SV2] 431: GSys EndP
[4U2] 432: Code EndS
[RV2] 433: End Start
[DK0] 434: End

```

Listing 2. Teil 2 des Assembler-Listings für »hercules.com«

```

[DU7] 0: E9 81 0B FB 53 51 52 57 56 1E 06 FC 0E 1F 8E 06
[D18] 1: FF 0B 8A 1E 1A 0C 88 1E 1B 0C C6 06 1A 0C 00 80
[YX7] 2: 3E 19 0C 00 74 03 E8 72 0A 80 FC 24 73 0A 8A DC
[B58] 3: B7 00 D1 E3 FF 97 24 0C 2E 8A 1E 1B 0C 2E 88 1E
[YZ7] 4: 1A 0C 07 1F 5E 5F 5A 59 5B CF 3C 03 77 0C 80 28
[B28] 5: BE 78 0C C6 06 FE 0B 00 EB 0A BO 0A BE 6C 0C C6
[SF8] 6: 06 FE 0B FF BA B8 03 EE B4 00 B2 B4 8A C4 EE AC
[AD8] 7: 42 EE 4A FE C4 80 FC 0C 75 F2 C3 A0 FE 0B C3 0A
[2R7] 8: D2 75 06 C6 06 1B 0C 00 C3 C6 06 1B 0C FF 49 88
[HN7] 9: 0E 14 0C 2B D1 88 16 16 0C C3 88 0E 10 0C 88 16
[MU7] 10: 12 0C C3 A0 10 0C 8A 26 12 0C C3 A2 01 0C C3 80
[568] 11: 3E FE 0B 00 74 0D BA B8 03 FE C8 75 03 80 0A B9
[568] 12: 0C 8A EE C3 B4 B0 FE C8 74 02 B4 B8 88 26 00 0C
[827] 13: C3 03 0E 08 0C A1 0C 0C 03 C2 E8 08 0A 8B F9 D1
[E28] 14: EF D1 EF D1 EF 03 F8 80 E1 07 B0 80 D2 E8 26 22
[KR7] 15: 05 74 02 B0 01 C3 03 0E 08 0C 03 16 0C 0C 3B 0E
[1X7] 16: 08 0C 72 41 3B 16 0C 0C 72 3B 3B 0E 0A 0C 77 35
[C48] 17: 3B 16 0E 0C 77 2F 8B C2 E8 CA 09 8B F9 D1 EF D1
[138] 18: EF D1 EF 03 F8 80 E1 07 B0 80 D2 E8 8A 26 01 0C
[2W7] 19: 0A E4 75 04 26 08 05 C3 FE CC 75 06 F6 0D 26 20
[CL7] 20: 05 C3 26 30 05 C3 03 0E 08 0C 03 3E 08 0C 03 16
[TZ7] 21: 0C 0C 03 36 0C 0C 3B F9 73 04 87 F9 87 F2 89 0E
[EX7] 22: 2F 0D 89 3E 33 0D C7 06 3B 0D 00 00 2B F9 89 3E
[238] 23: 37 0D 75 06 C7 06 3B 0D FF FF C7 06 3D 0D 01 00
[N18] 24: 3B F2 77 06 C7 06 3D 0D FF FF 89 16 31 0D 89 36
[BZ7] 25: 35 0D 2B D6 79 02 F7 DA 89 16 39 0D D1 06 02 0C
[V78] 26: 73 0B 8B 0E 2F 0D 8B 16 31 0D E8 51 FF 83 3E 3B
[1S7] 27: 0D 00 79 19 A1 35 0D 3B 06 31 0D 74 26 A1 3D 0D
[1W7] 28: 01 06 31 0D A1 37 0D 01 06 3B 0D EB CF A1 33 0D
[NZ7] 29: 3B 06 2F 0D 74 0D FF 06 2F 0D A1 39 0D 29 06 3B
[C28] 30: 0D EB B9 C3 89 0E 02 0C C3 89 0E 08 0C 89 3E 0A
[C38] 31: 0C 89 16 0C 0C 89 36 0E 0C C3 FD 8A D0 E8 FF 06
[668] 32: A1 0C 0C 8B 1E 08 0C 8B 3E 0A 0C D1 EB D1 EB D1
[658] 33: EB D1 EF D1 EF D1 EF 50 53 57 E8 C8 08 03 D8 03
[HY7] 34: F8 F7 D1 26 20 0F 26 20 2D 8A C2 F7 D1 22 C1 26
[1T7] 35: 08 07 8A C2 22 C5 26 08 05 8A C2 4F 3B DF 74 03
[E98] 36: AA EB F9 5F 5B 58 40 3B 06 0E 0C 76 CA C3 3C 0D
[B77] 37: 75 06 C6 06 10 0C 0C C3 8B 0E 10 0C 3C 08 75 09
[N28] 38: 0A C9 74 04 FE 0E 10 0C C3 D1 E1 D1 E1 D1 E1 03
[KY7] 39: 0E 08 0C 8B 16 12 0C D1 E2 D1 E2 D1 E2 03 16 0C
[NT7] 40: 0C 3C 0A 74 1A 51 52 E8 A6 06 5A 59 83 C1 10 3B
[BL7] 41: 0E 0A 0C 77 05 FE 06 10 0C C3 C6 06 10 0C 00 83
[PU7] 42: C2 10 3B 16 0E 0C 77 05 FE 06 12 0C C3 B2 08 EB
[GW7] 43: 42 03 0E 08 0C 03 16 0C 0C 83 C2 08 3B 16 0E 0C
[V28] 44: 77 2E 83 EA 08 8B DC 36 8E 5F 04 AC 8A E0 83 C1
[OT7] 45: 08 2E 3B 0E 0A 0C 77 18 83 E9 08 AC 50 51 52 56
[R98] 46: 1E E8 4C 06 1F 5E 5A 59 5B 83 C1 08 FE C1 75 DE
[B18] 47: C3 8A D0 B6 00 E8 07 06 8B 1E 08 0C D1 EB D1 EB
[A78] 48: D1 EB 89 1E 2F 0D A1 0C 0C 8B F0 03 F2 52 8B 16
[GD8] 49: 0A 0C D1 EA D1 EA D1 EA 2B D3 50 E8 C7 07 8B F8
[AB8] 50: 03 3E 2F 0D 8B C6 E8 BC 07 8B D8 03 1E 2F 0D F7
[FY7] 51: D1 26 20 0D F7 D1 26 8A 07 22 C1 26 05 03 DA
[BX7] 52: 03 FA F7 D1 26 20 2D F7 D1 26 8A 07 22 C5 26 08
[NA8] 53: 05 8A F2 4F FD 4B 26 8A 07 AA FE CE 75 F7 58 40
[LO8] 54: 46 3B 36 0E 0C 76 B3 5E A1 0E 0C 2B C6 50 E8 74
[008] 55: 07 8B F8 03 3E 2F 0D 2E 80 3E 01 0C 01 75 0C 26
[YZ7] 56: 08 0D 03 FA 26 08 2D B0 FF EB 0E F7 D1 26 20 0D
[F18] 57: 03 FA 26 20 2D F7 D1 B0 00 51 8B CA 49 4F F3 AA
[SN7] 58: 59 58 40 3B 06 0E 0C 76 CA C3 03 0E 08 0C 03 16
[V28] 59: 0C 0C 8B DF 8A DF B7 00 D1 E3 D1 E3 D1 E3 03 D3
[168] 60: 3B 16 0E 0C 77 3A 2B D3 8B DC 36 8E 5F 04 AC 8A
[F78] 61: E0 8B DF B7 00 D1 E3 D1 E3 D1 E3 03 CB 2E 3B 0E
[X18] 62: 0A 0C 77 1C 2B CB 53 8B DF AC 50 51 52 57 56 1E
[U58] 63: E8 D6 05 1F 5E 5F 5A 59 5B 58 03 CB FE C1 75 D1
[E08] 64: C3 03 0E 08 0C 03 16 0C 0C 8B 0C BF 65 17 89
[DY7] 65: 0C 89 54 02 83 C6 04 4E 4E 8B 04 4E 4E 8B 0C 49
[1M7] 66: E8 38 00 41 41 E8 33 00 49 48 E8 2E 00 40 40 E8
[4Y7] 67: 29 00 81 FE 84 0C 74 06 81 FE 65 17 75 D9 8B F7
[F68] 68: 81 FF 65 17 72 05 BF 84 0C EB 03 BF 65 17 81 FE
[CV7] 69: 84 0C 74 06 81 FE 65 17 75 BD C3 50 51 3B 06 0C
[8V7] 70: 0C 72 57 3B 06 0E 0C 77 51 3B 0E 08 0C 72 4B 3B
[N58] 71: 0E 0A 0C 77 45 E8 6D 06 8B D9 D1 EB D1 EB D1 EB
[NZ7] 72: 03 D8 B0 80 80 E1 07 D2 E8 8A E0 80 3E 01 0C 01
[MG7] 73: 74 13 26 22 07 75 23 26 08 27 59 58 89 0D 47 75
[GN7] 74: 89 05 47 47 C3 26 22 07 74 10 F6 D4 26 20 27 59
[KN7] 75: 58 89 0D 47 47 89 05 47 47 C3 59 58 C3 33 0C 80
[MW7] 76: 3E 01 0C 01 75 02 F7 D0 B9 00 40 33 FF F3 AB C3

```



```

[BW7] 77: 8C CO 8E D8 80 F4 08 8E CO B9 00 40 33 FF 33 F6
[G18] 78: F3 A5 C3 8C CO 80 F4 08 8E D8 B9 00 40 33 FF 33
[AA8] 79: F6 AD 26 8B 1D AB 89 5C FE E2 F6 C3 E8 00 4A 8B
[Y68] 80: 1E 08 0C D1 EB D1 EB D1 EB 8B 3E 0A OC D1 EF D1
[U08] 81: EF D1 EF A1 OC OC 50 53 57 E8 C9 05 03 D8 03 F8
[LW7] 82: 26 30 OF 26 30 2D 4F 3B FB 74 05 26 F6 15 EB F6
[768] 83: 5F 5B 58 40 3B 06 0E OC 76 DC C3 50 3B 16 OC OC
[GV7] 84: 77 1C C7 06 3D OD 01 00 89 16 31 OD A1 OC OC A3
[D58] 85: 35 OD 8B 1E 0E OC 2B D8 89 1E 39 OD EB 18 C7 06
[M38] 86: 3D OD FF FF A1 0E OC A3 35 OD 2B 06 OC OC A3 39
[708] 87: OD 03 C2 A3 31 OD FF 06 39 OD C6 06 3B OD 01 3B
[377] 88: 0E 08 OC 72 05 C6 06 3B OD 00 8B C1 D1 E8 D1 E8
[558] 89: D1 E8 A3 2F OD A1 08 OC D1 E8 D1 E8 D1 E8 A3 33
[B98] 90: OD 8B 1E 0A OC D1 EB D1 EB D1 EB 2B D8 4B 89 1E
[927] 91: 37 OD 8A D9 80 E3 07 8A 97 5C OC 8B D9 2B 1E 08
[FU7] 92: OC 03 1E 0A OC 81 E3 07 00 8A B7 64 OC 80 E1 07
[ER7] 93: AO 08 OC 24 07 2A C8 80 E1 OF C6 06 3C OD 00 80
[558] 94: F9 08 72 0A C6 06 3C OD 01 F6 D9 80 C1 10 58 BB
[477] 95: 00 B0 3C 01 74 02 B7 88 8E D8 2E A1 31 OD E8 EA
[M78] 96: 04 8B D8 2E A1 35 OD E8 D8 04 8B F8 2E 80 3E 3B
[X18] 97: OD 00 74 3B 2E 03 1E 2F OD 2E 03 3E 33 OD E8 F6
[LY7] 98: 02 F6 D2 20 17 47 F6 D2 22 E2 08 27 2E 8A 2E 37
[X78] 99: OD 43 0A ED 74 0A E8 DE 02 47 88 27 FE CD EB F1
[408] 100: E8 D4 02 F6 D6 20 37 F6 D6 22 E6 08 27 E8 45 2E
[ZX7] 101: 03 1E 2F OD 2E 03 1E 37 OD 43 2E 03 3E 33 OD 2E
[G98] 102: 03 3E 37 OD 47 E8 AF 02 F6 D6 20 37 4F F6 D6 22
[BY7] 103: E6 08 27 2E 8A 2E 37 OD 4B 0A ED 74 0A E8 97 02
[968] 104: 4F 88 27 FE CD EB F1 E8 8D 02 F6 D2 20 17 F6 D2
[DX7] 105: 22 E2 08 27 2E FF 0E 39 OD 74 1D 2E A1 31 OD 2E
[OX7] 106: 03 06 3D OD 2E A3 31 OD 2E A1 35 OD 2E 03 06 3D
[G98] 107: OD 2E A3 35 OD E9 42 FF C3 3B CF 76 02 87 CF 3B
[NS7] 108: D6 76 02 87 D6 03 0E 08 OC 03 16 OC OC 03 3E 08
[CV7] 109: OC 03 36 OC 0C 3B 0E 08 OC 73 04 8B 0E 08 OC 3B
[A08] 110: 0E 0A OC 77 D3 3B 3E 08 OC 72 CD 3B 3E 0A OC 76
[XW7] 111: 04 8B 3E 0A OC 3B 16 OC OC 73 04 8B 16 OC OC 3B
[4X7] 112: 16 0E OC 77 B3 3B 36 OC OC 72 AD 3B 36 0E OC 76
[W08] 113: 04 8B 36 0E OC 8B DF 81 E3 07 00 8A A7 64 OC 8A
[PC8] 114: D9 80 E3 07 8A 87 5C OC D1 E9 D1 E9 D1 E9 D1 EF
[KB8] 115: D1 EF D1 EF 2B F9 74 42 50 8B C2 E8 A7 03 8B D8
[857] 116: 03 D9 58 F7 D0 26 20 07 03 DF 26 20 27 F7 D0 53
[RZ7] 117: 8B DA 81 E3 07 00 8A AF 1C OC 5B 51 22 EC 26 08
[CV7] 118: 2F 59 57 48 4F 74 05 26 88 2F E8 F7 22 E8 26 08
[N18] 119: 2F 5F B5 00 42 3B D6 76 BF C3 22 C4 8B DA 81 E3
[Z27] 120: 07 00 8A A7 1C OC 50 8B C2 E8 59 03 8B D8 03 D9
[TX7] 121: 58 F6 D0 26 20 07 F6 D0 22 E0 26 08 27 42 3B D6
[158] 122: 76 DA C3 89 0E 1C OC 89 16 1E OC 89 3E 20 OC 89
[V58] 123: 36 22 OC C3 8B DF 8B F3 D1 EB D1 EB 2B F3 03 0E
[XU7] 124: 08 OC 03 16 OC OC 89 0E 2F OD 89 16 31 OD C7 06
[408] 125: 33 OD 00 00 89 36 35 OD 57 4F 8B C6 F7 E6 A3 3F
[7P7] 126: OD 89 16 41 OD C7 06 43 OD 00 00 D1 E0 D1 D2 A3
[958] 127: 55 OD 89 16 57 OD 8B C7 F7 E7 A3 4B OD 89 16 4D
[6W7] 128: OD 50 52 D1 E0 D1 D2 A3 51 OD 89 16 53 OD 58 8B
[R48] 129: DE D1 E3 4B F7 E3 A3 47 OD 89 16 49 OD 58 F7 E3
[UC7] 130: A3 45 OD 01 16 47 OD 83 16 49 OD 00 A1 3F OD 01
[DP7] 131: 06 4B OD A1 41 OD 11 06 4D OD C7 06 4F OD 00 00
[YN7] 132: 5E 8B 0E 2F OD 8B 16 31 OD 03 0E 33 OD 03 16 35
[X58] 133: OD E8 AA F8 8B 0E 2F OD 8B 16 31 OD 2B 0E 33 OD
[YW7] 134: 03 16 35 OD E8 97 F8 8B 0E 2F OD 8B 16 31 OD 03
[C27] 135: 0E 33 OD 2B 16 35 OD E8 84 F8 8B 0E 2F OD 8B 16
[SU7] 136: 31 OD 2B 0E 33 OD 2B 16 35 OD E8 71 F8 F6 06 50
[MZ7] 137: OD 80 75 2E FF 06 33 OD A1 3F OD 29 06 4B OD A1
[KU7] 138: 41 OD 19 06 4D OD A1 43 OD 19 06 4F OD A1 55 OD
[6F7] 139: 01 06 3F OD A1 57 OD 11 06 41 OD 83 16 43 OD 00
[4Y7] 140: EB 2C FF 0E 35 OD A1 45 OD 01 06 4B OD A1 47 OD
[GL7] 141: 11 06 4D OD A1 49 OD 11 06 4F OD A1 51 OD 29 06
[OX7] 142: 45 OD A1 53 OD 19 06 47 OD 83 1E 49 OD 00 83 3E
[TR7] 143: 35 OD 00 78 09 39 36 33 OD 77 03 E9 43 FF C3 8B
[9W7] 144: 1E 08 OC 81 E3 07 00 8A 8F 5C OC 8A 1E 0A OC 80
[RR7] 145: E3 07 8A AF 64 OC C3 26 8A 25 2E 80 3E 3C OD 00
[PS7] 146: 75 07 26 8A 45 FF D3 C8 C3 26 8A 45 01 D3 C0 C3
[MX7] 147: A8 80 74 10 33 DB 8E DB 8B 36 7C 00 8E 1E 7E 00
[CD8] 148: 24 7F EB 08 BB 00 F0 8E DB BE 6E FA B4 00 D1 E0
[ZA8] 149: D1 E0 D1 E0 03 F0 8B F9 D1 EF D1 EF D1 EF 80 E1
[FV7] 150: 07 B5 08 8B C2 E8 7D 01 57 03 F8 AC 8A E0 B0 00
[KW7] 151: D3 E8 86 E0 2E 8A 3E 01 OC 0A FF 75 05 26 09 05
[S57] 152: EB 0E FE CF 75 07 F7 D0 26 21 05 EB 03 26 31 05
[B58] 153: 5F 42 FE CD 75 CD 0E 1F C3 1E 0E 1F 80 3E FE 0B
[VW7] 154: 00 74 20 80 3E 1A OC 00 74 19 FB FE 0E 18 OC 75
[558] 155: 12 C6 06 18 OC 05 53 06 8B 1E FF 0B 8E C3 E8 DA
[3X7] 156: 00 07 5B 1F 2E FF 2E 04 OC 53 2E 89 0E 2F OD 2E
[1W7] 157: 89 16 31 OD A8 80 74 10 33 DB 8E DB 8B 36 7C 00
[C98] 158: 8E 1E 7E 00 24 7F EB 08 BB 00 F0 8E DB BE 6E FA
[Z18] 159: B4 00 D1 E0 D1 E0 03 F0 5B BA 00 00 8A CB
[357] 160: B5 00 0A F2 D0 E2 FA B2 00 2E C6 06 35 OD 08
[W38] 161: 2E 88 3E 3B OD 2E A1 31 OD 2E FF 36 2F OD E8 C4
[DV7] 162: 00 50 8A 04 2E C6 06 33 OD 08 D0 C0 73 39 2E 8B
[938] 163: 0E 2F OD 80 E1 07 5F 57 52 D3 CA 86 F2 2E 8A 0E
[158] 164: 2F OD D1 E9 D1 E9 D1 E9 03 F9 2E 8A 26 01 OC 0A
[TX7] 165: E4 75 05 26 09 15 EB 0E FE CC 75 07 F7 D2 26 21
[7U7] 166: 15 EB 03 26 31 15 5A 2E 00 1E 2F OD 2E 80 16 30
[S77] 167: OD 00 2E FE 0E 33 OD 75 B1 58 2E 8F 0E 2F OD 2E
[M28] 168: FF 06 31 OD 2E FE 0E 3B OD 75 8A 2E 88 3E 3B OD
[L58] 169: 46 2E FE 0E 35 OD 74 4C E9 7A FF 50 51 52 8B 1E
[N38] 170: 08 OC 8A CB D1 EB D1 EB D1 EB 03 1E 10 OC 80 E1
[Z28] 171: 07 BA 00 FF D3 EA 86 F2 8B 0E 16 OC A1 12 OC D1
[RK7] 172: E0 D1 E0 D1 E0 03 06 OC 03 06 14 OC 50 53 E8
[KR7] 173: 13 00 03 D8 26 31 17 5B 58 40 E2 F1 80 36 19 OC
[B48] 174: FF 5A 59 58 C3 53 8B D8 D1 E8 D1 E8 8A 5A F6 EA
[CV7] 175: 81 E3 03 00 D1 CB D1 CB D1 CB 03 C3 5B C3 00 00
[3G7] 176: B0 00 FF FF 00 00 00 00 00 00 CF 02 00 00 5B 01
[A97] 177: 00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00
[3K7] 178: 00 00 00 00 4A 01 7B 01 7F 01 9A 01 A3 01 AB 01
[FU7] 179: AF 01 C4 01 BD 05 D0 05 E3 05 4E 03 B1 03 AA 04

```

```

[3U7] 180: F1 03 F6 01 D1 01 E4 02 46 02 B4 08 BE 08 01 05
[G48] 181: B9 07 A3 08 E9 02 FA 02 FC 05 3B 06 FF 7F 3F 1F
[EZ7] 182: 0F 07 03 01 80 C0 E0 F0 F8 FC FE FF 35 2D 2E 07
[KD7] 183: 5B 02 57 57 02 03 00 00 61 50 52 0F 19 06 19 19
[4V7] 184: 02 OD 0B OC 8B 1C 35 CD 21 89 1E 04 OC 8C 06 06
[N88] 185: OC BE 99 0A 8B FB B9 2A 00 F3 A6 E3 1B BA 99 0A
[MV7] 186: B8 1C 25 CD 21 BA 03 01 B0 60 CD 21 BA C1 OC B4
[Z27] 187: 09 CD 21 BA 46 22 CD 27 BA 01 OD B4 09 CD 21 CD
[EM7] 188: 20 47 72 61 66 69 6B 2D 54 72 65 69 62 65 72 20
[PN7] 189: 66 81 72 20 48 65 72 63 75 6C 65 73 2D 4B 61 72
[MG7] 190: 74 65 20 56 20 31 2E 31 20 20 28 43 29 20 46 2E
[8S7] 191: 20 42 72 65 6E 64 6C 65 20 31 39 38 37 2E OD OA
[2K7] 192: 24 44 65 72 20 47 72 61 66 69 6B 2D 54 72 65 69
[4C7] 193: 62 65 72 20 69 73 74 20 62 65 72 65 69 74 73 20
[UP7] 194: 69 6E 73 74 61 6C 6C 69 65 72 74 21 0A OD 24 00
[007] 195: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 196: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 197: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 198: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 199: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 201: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 202: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 203: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 204: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 205: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 206: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 207: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 208: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 209: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[007] 210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

Listing 3. Für die hexadezimale Eingabe des Programms »hercules.com« brauchen Sie keinen Assembler

```

[W12] 0: Page 60,132
[000] 1:
[MX6] 2: Title Hercules-Treiber
[000] 3:
[FBH] 4: Video Equ 60H ; In
; Interrupt zum Aufrufen der Grafik-
[S63] 5: ; Ro
; utinen.
[000] 6:
[684] 7: CSeg Segment
[4T4] 8: Assume CS:CSeg
[000] 9:
[8X4] 10: Driver Proc Near
[000] 11:
[N2C] 12: ; Video-Controller auf Grafik schal
; ten.
[4XE] 13: Hires: Mov AX,4 ; Un
; terprogramm aufrufen.
[MU2] 14: Int Video
[SL0] 15: Ret
[000] 16:
[48B] 17: ; Video-Controller auf Text schalte
; n.
[DLB] 18: Text: XOr AX,AX ; AX
; =0 --> Textmodus.
[44D] 19: Int Video ; Un
; terprogramm aufrufen.
[SL0] 20: Ret
[000] 21:
[3K8] 22: ; Cursor-Position setzen.
[Y9E] 23: GotoXY: Mov AH,3 ; Fu
; nktions-Code nach AH.
[3F7] 24: Push BP ; BP
; retten.
[1FM] 25: Mov BP,SP ; SP
; nach BP schaffen (um auf den Stack zuzugrei
; fen).
[5RD] 26: Mov CX,Word Ptr SS:[BP+6] ; X-
; Koordinate nach CX.
[AF1] 27: Dec CX
[2RD] 28: Mov DX,Word Ptr SS:[BP+4] ; Y-
; Koordinate nach DX.
[BF1] 29: Dec DX
[DLA] 30: Pop BP ; BP
; wiederherstellen.
[44D] 31: Int Video ; Un
; terprogramm aufrufen.
[SL6] 32: Ret 4 ; Fe
; rtig.
[000] 33:
[5A9] 34: ; Aussehen des Cursors setzen.

```


[RTE]	35:	SetCurs: Mov	AH,2		; Fu
[3F7]	36:	nktions-Code nach	AH.		
		Push	BP		; BP
[1FM]	37:	retten.			
		Mov	BP,SP		; SP
		nach BP schaffen (um auf den Stack zuzugrei			fen).
[Z3C]	38:	Mov	CX,Word Ptr SS:[BP+6]	; 1.	
		Zeile nach CX.			
[AED]	39:	Mov	DX,Word Ptr SS:[BP+4]	; Le	
		tzte Zeile nach DX.			
[DLA]	40:	Pop	BP		; BP
		wiederherstellen.			
[44D]	41:	Int	Video		; Un
		terprogramm aufrufen.			
[SL6]	42:	Ret	4		; Fe
		rtig.			
[000]	43:				
[627]	44:				
[UME]	45:	SetMode: Mov	AH,5		; Fu
		nktions-Code nach	AH.		
[HFM]	46:	Mov	BX,SP		; SP
		nach BP schaffen (um auf den Stack zuzugrei			fen).
[E6E]	47:	Mov	AL,Byte Ptr SS:[BX+2]	; Ze	
		ichen-Modus nach AL.			
[44D]	48:	Int	Video		; Un
		terprogramm aufrufen.			
[PL6]	49:	Ret	2		; Fe
		rtig.			
[000]	50:				
[VW6]	51:				
[XMD]	52:	Disp: Mov	AH,6		; Fu
		nktions-Code nach	AH.		
[HFM]	53:	Mov	BX,SP		; SP
		nach BP schaffen (um auf den Stack zuzugrei			fen).
[TVD]	54:	Mov	AL,Byte Ptr SS:[BX+2]	; Se	
		itennummer nach AL.			
[44D]	55:	Int	Video		; Un
		terprogramm aufrufen.			
[PL6]	56:	Ret	2		; Fe
		rtig.			
[000]	57:				
[6ED]	58:				
		; Seite, auf die gezeichnet wird,			
		wählen.			
[WLE]	59:	SetPage: Mov	AH,7		; Fu
		nktions-Code nach	AH.		
[HFM]	60:	Mov	BX,SP		; SP
		nach BP schaffen (um auf den Stack zuzugrei			fen).
[TVD]	61:	Mov	AL,Byte Ptr SS:[BX+2]	; Se	
		itennummer nach AL.			
[44D]	62:	Int	Video		; Un
		terprogramm aufrufen.			
[PL6]	63:	Ret	2		; Fe
		rtig.			
[000]	64:				
[U08]	65:				
[7UE]	66:	ScrollUp: Mov	AH,14		; Fu
		nktions-Code nach	AH.		
[HFM]	67:	Mov	BX,SP		; SP
		nach BP schaffen (um auf den Stack zuzugrei			fen).
[4FD]	68:	Mov	AL,Byte Ptr SS:[BX+2]	; Ze	
		ilenzahl nach AL.			
[44D]	69:	Int	Video		; Un
		terprogramm aufrufen.			
[PL6]	70:	Ret	2		; Fe
		rtig.			
[000]	71:				
[OUA]	72:				
[M3E]	73:	Write: Mov	AH,11		; Fu
		nktions-Code nach	AH.		
[HFM]	74:	Mov	BX,SP		; SP
		nach BP schaffen (um auf den Stack zuzugrei			fen).
[EDC]	75:	Mov	AL,Byte Ptr SS:[BX+2]	; Ze	
		ichen nach AL.			
[44D]	76:	Int	Video		; Un
		terprogramm aufrufen.			
[PL6]	77:	Ret	2		; Fe
		rtig.			
[000]	78:				
[3B5]	79:				
[K5E]	80:	Point: Mov	AH,15		; Fu
		nktions-Code nach	AH.		
[3F7]	81:	Push	BP		; BP
		retten.			
[1FM]	82:	Mov	BP,SP		; SP
		nach BP schaffen (um auf den Stack zuzugrei			fen).
[5RD]	83:	Mov	CX,Word Ptr SS:[BP+6]	; X-	
		Koordinate nach CX.			
[2RD]	84:	Mov	DX,Word Ptr SS:[BP+4]	; Y-	
		Koordinate nach DX.			
[DLA]	85:	Pop	BP		; BP
		wiederherstellen.			
[44D]	86:	Int	Video		; Un
		terprogramm aufrufen.			
[SL6]	87:	Ret	4		; Fe
		rtig.			
[000]	88:				
[UF5]	89:				
[1RD]	90:	Line: Mov	AH,18		; Fu
		nktions-Code nach	AH.		
[3F7]	91:	Push	BP		; BP
		retten.			
[1FM]	92:	Mov	BP,SP		; SP
		nach BP schaffen (um auf den Stack zuzugrei			fen).
[S3E]	93:	Mov	CX,Word Ptr SS:[BP+10]	; X1	
		-Koordinate nach CX.			
[DOE]	94:	Mov	DX,Word Ptr SS:[BP+8]	; Y1	
		-Koordinate nach DX.			
[XWD]	95:	Mov	DI,Word Ptr SS:[BP+6]	; X2	
		-Koordinate nach DI.			
[1ZD]	96:	Mov	SI,Word Ptr SS:[BP+4]	; Y2	
		-Koordinate nach SI.			
[DLA]	97:	Pop	BP		; BP
		wiederherstellen.			
[44D]	98:	Int	Video		; Un
		terprogramm aufrufen.			
[WL6]	99:	Ret	8		; Fe
		rtig.			
[000]	100:				
[3R6]	101:				
[YME]	102:	Window: Mov	AH,24		; Fu
		nktions-Code nach	AH.		
[3F7]	103:	Push	BP		; BP
		retten.			
[1FM]	104:	Mov	BP,SP		; SP
		nach BP schaffen (um auf den Stack zuzugrei			fen).
[S3E]	105:	Mov	CX,Word Ptr SS:[BP+10]	; X1	
		-Koordinate nach CX.			
[DOE]	106:	Mov	DX,Word Ptr SS:[BP+8]	; Y1	
		-Koordinate nach DX.			
[XWD]	107:	Mov	DI,Word Ptr SS:[BP+6]	; X2	
		-Koordinate nach DI.			
[1ZD]	108:	Mov	SI,Word Ptr SS:[BP+4]	; Y2	
		-Koordinate nach SI.			
[DLA]	109:	Pop	BP		; BP
		wiederherstellen.			
[44D]	110:	Int	Video		; Un
		terprogramm aufrufen.			
[WL6]	111:	Ret	8		; Fe
		rtig.			
[000]	112:				
[BD7]	113:				
[W7E]	114:	Invert: Mov	AH,26		; Fu
		nktionscode nach	AH.		
[44D]	115:	Int	Video		; Un
		terprogramm aufrufen.			
[SL0]	116:	Ret			
[000]	117:				
[SDA]	118:				
[DUE]	119:	FillWin: Mov	AH,25		; Fu
		nktions-Code nach	AH.		
[HFM]	120:	Mov	BX,SP		; SP
		nach BP schaffen (um auf den Stack zuzugrei			fen).
[UYC]	121:	Mov	AL,Byte Ptr SS:[BX+2]	; Bi	
		tmuster nach AL.			
[44D]	122:	Int	Video		; Un
		terprogramm aufrufen.			
[PL6]	123:	Ret	2		; Fe
		rtig.			
[000]	124:				
[8D9]	125:				
[ZLE]	126:	ClearScr: Mov	AH,8		; Fu
		nktions-Code nach	AH.		
[44D]	127:	Int	Video		; Un
		terprogramm aufrufen.			
[SL0]	128:	Ret			
[000]	129:				
[TS9]	130:				
[DEF]	131:	GetMode: Mov	AH,1		; Un
		terprogramm aufrufen.			
[MU2]	132:	Int	Video		
[ZC2]	133:	Mov	BX,SP		
[6L9]	134:	Or	AL,AL		; Wa
		hr oder Falsch?			
[VY3]	135:	Jnz	GetModel		
[CXC]	136:	Mov	Byte Ptr SS:[BX+2],1	; Gr	
		afikmodus aktiv.			
[AB1]	137:	Ret	2		
[E3D]	138:	GetModel: Mov	Byte Ptr SS:[BX+2],0	; Te	
		xtmodus.			
[AB1]	139:	Ret	2		
[000]	140:				


```

[38D] 141:      ; Zeichenkette auf dem Bildschirm
         ausgeben.
[R4E] 142: String: Mov  AH,12      ; Fu
         nktionscode nach AH.
[LF7] 143:      Push  DS          ; DS
         retten.
[RDA] 144:      Push  SS          ; SS
         nach DS schaffen.
[8H1] 145:      Pop    DS
[4FM] 146:      Mov    SI,SP      ; SP
         nach BP schaffen (um auf den Stack zuzugrei
         fen).
[TUD] 147:      Mov    CX,Word Ptr DS:[SI+97]; X-
         Koordinate nach CX.
[MUD] 148:      Mov    DX,Word Ptr DS:[SI+95]; Y-
         Koordinate nach DX.
[5LA] 149:      Add    SI,4      ; St
         ring-Anfang nach SI.
[44D] 150:      Int    Video     ; Un
         terprogramm aufrufen.
[XLA] 151:      Pop    DS        ; DS
         wiederherstellen.
[7N6] 152:      Ret     95       ; Fe
         rtig.
[000] 153:      ; Figur füllen.
[V75] 154:      FillShp: Mov  AH,21      ; Fu
[9TE] 155:      nktions-Code nach AH.
[3F7] 156:      Push  BP        ; BP
         retten.
[1FM] 157:      Mov    BP,SP      ; SP
         nach BP schaffen (um auf den Stack zuzugrei
         fen).
[5RD] 158:      Mov    CX,Word Ptr SS:[BP+6]; X-
         Koordinate nach CX.
[2RD] 159:      Mov    DX,Word Ptr SS:[BP+4]; Y-
         Koordinate nach DX.
[DLA] 160:      Pop    BP        ; BP
         wiederherstellen.
[44D] 161:      Int    Video     ; Un
         terprogramm aufrufen.
[SL6] 162:      Ret     4        ; Fe
         rtig.
[000] 163:      ; Window kopieren.
[1SS] 164:      CopyWin: Mov  AH,27      ; Fu
[2XE] 165:      nktions-Code nach AH.
[3F7] 166:      Push  BP        ; BP
         retten.
[1FM] 167:      Mov    BP,SP      ; SP
         nach BP schaffen (um auf den Stack zuzugrei
         fen).
[1WD] 168:      Mov    AL,Byte Ptr SS:[BP+8]; Se
         itennummer nach AL.
[ZZD] 169:      Mov    CX,Word Ptr SS:[BP+6]; X1
         -Koordinate nach CX.
[XZD] 170:      Mov    DX,Word Ptr SS:[BP+4]; Y1
         -Koordinate nach DX.
[DLA] 171:      Pop    BP        ; BP
         wiederherstellen.
[44D] 172:      Int    Video     ; Un
         terprogramm aufrufen.
[UL6] 173:      Ret     6        ; Fe
         rtig.
[000] 174:      ; Großschrift ausgeben.
[Z98] 175:      PrintB: Mov  AH,13      ; Fu
[KCE] 176:      nktions-Code nach AH.
[LF7] 177:      Push  DS          ; DS
         retten.
[RDA] 178:      Push  SS          ; SS
         nach DS schaffen.
[8H1] 179:      Pop    DS
[KC2] 180:      Mov    SI,SP
[C1E] 181:      Mov    CX,Word Ptr DS:[SI+101]; X
         -Koordinate nach CX.
[8VD] 182:      Mov    DX,Word Ptr DS:[SI+99]; Y-
         Koordinate nach DX.
[08E] 183:      Mov    BL,Byte Ptr DS:[SI+97]; X-
         Vergrößerung holen.
[F7E] 184:      Mov    BH,Byte Ptr DS:[SI+95]; Y-
         Vergrößerung holen.
[2ME] 185:      Mov    DI,BX      ; Ve
         rgrößerungswerte nach DI.
[ET1] 186:      Add    SI,4
[EKB] 187:      Int    Video     ; Fu
         nktion ausführen.
[XLA] 188:      Pop    DS        ; DS
         wiederherstellen.
[FN6] 189:      Ret     99       ; Fe
         rtig.
[000] 190:      ; Farbe eines Punktes ermitteln.
[2AA] 191:      GetCol: Mov  AH,16      ; Fu
[TEE] 192:      nktions-Code nach AH.

```

```

[NU1] 193:      Push  BP
[CKA] 194:      Mov    BP,SP      ; SP
         nach BP schaffen.
[5RD] 195:      Mov    CX,Word Ptr SS:[BP+6]; X-
         Koordinate nach CX.
[2RD] 196:      Mov    DX,Word Ptr SS:[BP+4]; Y-
         Koordinate nach DX.
[1GA] 197:      Int    Video     ; Fa
         rbe ermitteln.
[LW9] 198:      Mov    AH,0      ; Au
         f 16 Bit erweitern.
[FUC] 199:      Mov    Word Ptr SS:[BP+8],AX ; Fa
         rbe übergeben.
[OH1] 200:      Pop    BP
[EB1] 201:      Ret     6
[000] 202:      ; Kreis zeichnen.
[GK5] 203:      Circle: Mov  AH,19      ; Fu
[3F7] 204:      nktionscode nach AH.
[3F7] 205:      Push  BP        ; BP
         retten.
[1FM] 206:      Mov    BP,SP      ; SP
         nach BP schaffen (um auf den Stack zuzugrei
         fen).
[DRD] 207:      Mov    CX,Word Ptr SS:[BP+8]; X-
         Koordinate nach CX.
[ARD] 208:      Mov    DX,Word Ptr SS:[BP+6]; Y-
         Koordinate nach DX.
[PSB] 209:      Mov    DI,Word Ptr SS:[BP+4]; Ra
         dius nach DI.
[DLA] 210:      Pop    BP        ; BP
         wiederherstellen.
[44D] 211:      Int    Video     ; Un
         terprogramm aufrufen.
[UL6] 212:      Ret     6        ; Fe
         rtig.
[000] 213:      ; Ellipse zeichnen.
[SU5] 214:      Ellipse: Mov  AH,20      ; Fu
[ZEE] 215:      nktionscode nach AH.
[3F7] 216:      Push  BP        ; BP
         retten.
[1FM] 217:      Mov    BP,SP      ; SP
         nach BP schaffen (um auf den Stack zuzugrei
         fen).
[YUD] 218:      Mov    CX,Word Ptr SS:[BP+10]; X-
         Koordinate nach CX.
[KRD] 219:      Mov    DX,Word Ptr SS:[BP+8]; Y-
         Koordinate nach DX.
[A5D] 220:      Mov    DI,Word Ptr SS:[BP+6]; X-
         Halbachse nach DI.
[G6D] 221:      Mov    SI,Word Ptr SS:[BP+4]; Y-
         Halbachse nach SI.
[DLA] 222:      Pop    BP        ; BP
         wiederherstellen.
[44D] 223:      Int    Video     ; Un
         terprogramm aufrufen.
[WL6] 224:      Ret     8        ; Fe
         rtig.
[000] 225:      ; X-Koordinate des Cursors ermitte
         ln.
[K5F] 227:      WhereX: Mov  AH,4      ; Un
         terprogramm aufrufen.
[MU2] 228:      Int    Video
[NU1] 229:      Push  BP
[FC2] 230:      Mov    BP,SP
[LW9] 231:      Mov    AH,0      ; Au
         f 16 Bit erweitern.
[CHH] 232:      Inc    AX      ; X-
         Koordinate startet jetzt bei 1 statt bei 0.
[ERE] 233:      Mov    Word Ptr SS:[BP+4],AX ; Fu
         nktionswert ablegen.
[OH1] 234:      Pop    BP
[AB1] 235:      Ret     2
[000] 236:      ; Y-Koordinate des Cursors ermitte
         ln.
[S5F] 238:      WhereY: Mov  AH,4      ; Un
         terprogramm aufrufen.
[MU2] 239:      Int    Video
[NU1] 240:      Push  BP
[FC2] 241:      Mov    BP,SP
[POC] 242:      Mov    AL,AH      ; Y-
         Koordinate steht in AH!
[2V1] 243:      Mov    AH,0
[DDH] 244:      Inc    AX      ; Y-
         Koordinate startet jetzt bei 1 statt bei 0.
[ERE] 245:      Mov    Word Ptr SS:[BP+4],AX ; Fu
         nktionswert ablegen.
[OH1] 246:      Pop    BP
[AB1] 247:      Ret     2
[000] 248:      ; Bildschirminhalt in andere Seite
[ZND] 249:      kopieren.

```



```
[DCE] 250: ScrCopy: Mov     AH,9           ; Fu
           nktionscode nach AH.
[44D] 251: Int     Video           ; Un
           terprogramm aufrufen.
[SL0] 252: Ret
[000] 253:
[4EG] 254:           ; Inhalt von erster und zweiter Se
           ite vertauschen.
[X7D] 255: Swap: Mov     AH,10          ; Fu
           nktionscode nach AH.
[2UC] 256: Int     Video           ; Un
           terprogramm aufrufen
[SL0] 257: Ret
[000] 258:
[P6A] 259:           ; Aussehen der Linien festlegen.
[7KE] 260: SetStyle: Mov  AH,17         ; Fu
           nktionscode nach AH.
[CLA] 261: Mov     BX,SP              ; SP
           nach BX schaffen.
[F0D] 262: Mov     CX,Word Ptr SS:[BX+2] ; Bi
           tmuster nach CX.
[44D] 263: Int     Video           ; Un
           terprogramm aufrufen.
[AB1] 264: Ret     2
[000] 265:
[AY5] 266:           ; FillPattern Befehl.
[XKD] 267: FilPt:  Mov  AH,22          ; Fu
           nktionscode nach AH.
[3F7] 268: Push    BP                ; BP
           retten.
[1FM] 269: Mov     BP,SP              ; SP
           nach BP schaffen (um auf den Stack zuzugrei
           fen).
[S3E] 270: Mov     CX,Word Ptr SS:[BP+10]; X1
           -Koordinate nach CX.
[D0E] 271: Mov     DX,Word Ptr SS:[BP+8] ; Y1
           -Koordinate nach DX.
[XWD] 272: Mov     DI,Word Ptr SS:[BP+6] ; X2
           -Koordinate nach DI.
[1ZD] 273: Mov     SI,Word Ptr SS:[BP+4] ; Y2
           -Koordinate nach SI.
[DLA] 274: Pop     BP                ; BP
           wiederherstellen.
[44D] 275: Int     Video           ; Un
           terprogramm aufrufen.
[WL6] 276: Ret     8                ; Fe
           rtig.
[000] 277:
[RNA] 278:           ; Füllmuster für FillP definieren.
[7KD] 279: DefPt:  Mov  AH,23          ; Fu
           nktionscode nach AH.
[ZC2] 280: Mov     BX,SP
[V0G] 281: Mov     ES,Word Ptr SS:[BX+4] ; Se
           gment der Tabelle nach ES.
[OEF] 282: Mov     BX,Word Ptr SS:[BX+2] ; Of
           fset der Tabelle nach BX.
[P5G] 283: Mov     CX,Word Ptr ES:[BX]   ; Fü
           llmuster in Register laden.
[FX6] 284: Mov     DX,Word Ptr ES:[BX+2]
[TW6] 285: Mov     DI,Word Ptr ES:[BX+4]
[GX6] 286: Mov     SI,Word Ptr ES:[BX+6]
[44D] 287: Int     Video           ; Un
           terprogramm aufrufen.
[SL6] 288: Ret     4                ; Fe
           rtig.
[000] 289:
[MP3] 290: Driver  EndP
[RS2] 291: CSeg   EndS
[DK0] 292: End
```

Listing 4. Der Assembler-Quellcode für den Hercules-Treiber »hires.bin«

```
[2Y7] 0: B8 04 00 CD 60 C3 33 C0 CD 60 C3 B4 03 55 8B EC
[OX7] 1: 8B 4E 06 49 8B 56 04 4A 5D CD 60 C2 04 00 B4 02
[A58] 2: 55 8B EC 8B 4E 06 8B 56 04 5D CD 60 C2 04 00 B4
[M78] 3: 05 8B DC 36 8A 47 02 CD 60 C2 02 00 B4 06 8B DC
[F27] 4: 36 8A 47 02 CD 60 C2 02 00 B4 07 8B DC 36 8A 47
[F08] 5: 02 CD 60 C2 02 00 B4 0E 8B DC 36 8A 47 02 CD 60
[UL7] 6: C2 02 00 B4 0B 8B DC 36 8A 47 02 CD 60 C2 02 00
[028] 7: B4 0F 55 8B EC 8B 4E 06 8B 56 04 5D CD 60 C2 04
[M38] 8: 00 B4 12 55 8B EC 8B 4E 0A 8B 56 08 8B 7E 06 8B
[ZU7] 9: 76 04 5D CD 60 C2 08 00 B4 18 55 8B EC 8B 4E 0A
[B08] 10: 8B 56 08 8B 7E 06 8B 76 04 5D CD 60 C2 08 00 B4
[G68] 11: 1A CD 60 C3 B4 19 8B DC 36 8A 47 02 CD 60 C2 02
```

```
[ZP7] 12: 00 B4 08 CD 60 C3 B4 01 CD 60 8B DC 0A C0 75 08
[GA7] 13: 36 C6 47 02 01 C2 02 00 36 C6 47 02 00 C2 02 00
[HY7] 14: B4 0C 1E 16 1F 8B F4 8B 4C 61 8B 54 5F 83 C6 04
[M18] 15: CD 60 1F C2 5F 00 B4 15 55 8B EC 8B 4E 06 8B 56
[R28] 16: 04 5D CD 60 C2 04 00 B4 1B 55 8B EC 8A 46 08 8B
[6T7] 17: 4E 06 8B 56 04 5D CD 60 C2 06 00 B4 0D 1E 16 1F
[E68] 18: 8B F4 8B 4C 65 8B 54 63 8A 5C 61 8A 7C 5F 8B FB
[PX7] 19: 83 C6 04 CD 60 1F C2 63 00 B4 10 55 8B EC 8B 4E
[TU7] 20: 06 8B 56 04 CD 60 B4 00 89 46 08 5D C2 06 00 B4
[LY7] 21: 13 55 8B EC 8B 4E 08 8B 56 06 8B 7E 04 5D CD 60
[R18] 22: C2 06 00 B4 14 55 8B EC 8B 4E 0A 8B 56 08 8B 7E
[W18] 23: 06 8B 76 04 5D CD 60 C2 08 00 B4 04 CD 60 55 8B
[DM7] 24: EC B4 00 40 89 46 04 5D C2 02 00 B4 04 CD 60 55
[UT7] 25: 8B EC 8A C4 B4 00 40 89 46 04 5D C2 02 00 B4 09
[EW7] 26: CD 60 C3 B4 0A CD 60 C3 B4 11 8B DC 36 8B 4F 02
[LV7] 27: CD 60 C2 02 00 B4 16 55 8B EC 8B 4E 0A 8B 56 08
[H68] 28: 8B 7E 06 8B 76 04 5D CD 60 C2 08 00 B4 17 8B DC
[HR7] 29: 36 8E 47 04 36 8B 5F 02 26 8B 0F 26 8B 57 02 26
[AD6] 30: 8B 7F 04 26 8B 77 06 CD 60 C2 04 00 E4
```

Listing 5. Sie können »hires.bin« auch in hexadezimaler Schreibweise eingeben

```
[57F] 0: Const White=0; Black=1; Invers=2; XMaxGlb=7
           19; YMaxGlb=347;
[1M7] 1: Type MaxString=String[90];
[168] 2: Var ConOutPtrText: Integer;
[N7G] 3: Procedure Hires;
           External 'HIRES.BIN';
[03H] 4: Procedure TextMode;
           External Hires[0006];
[WON] 5: Procedure GotoXYGraphic(X,Y: Integer);
           External Hires[000B];
[U4R] 6: Procedure SetCursor(FirstLine,LastLine: Inte
           ger);
           External Hires[010E];
[ZEM] 7: Procedure HiResColor(Farbe: Integer);
           External Hires[020F];
[FUN] 8: Procedure DisplayScreen(Seite: Integer);
           External Hires[030C];
[3YM] 9: Procedure SelectScreen(Seite: Integer);
           External Hires[0409];
[14K] 10: Procedure ClearScreen;
           External Hires[0C01];
[KTH] 11: Procedure CopyScreen;
           External Hires[190E];
[7PH] 12: Procedure SwapScreen;
           External Hires[1A03];
[CZN] 13: Procedure WriteCharacter(Zeichen: Char);
           External Hires[1B03];
[WBP] 14: Procedure DrawString(X,Y: Integer; S: MaxStrin
           g);
           External Hires[1E00];
[2RR] 15: Procedure DrawText(X,Y,XVer,YVer: Integer; S:
           MaxString);
           External Hires[110B];
[WRM] 16: Procedure ScrollUp(Zeilen: Integer);
           External Hires[1056];
[OLK] 17: Procedure Plot(X,Y: Integer);
           External Hires[1070];
[XGP] 18: Procedure SetLineStyle(Bitmuster: Integer);
           External Hires[1A08];
[A9L] 19: Procedure Draw(X1,Y1,X2,Y2: Integer);
           External Hires[1081];
[SHM] 20: Procedure Circle(X,Y,Radius: Integer);
           External Hires[140F];
[C4T] 21: Procedure DrawEllipse(X,Y,XHalbAchse,YHalbA
           chse: Integer);
           External Hires[1603];
[B1M] 22: Procedure FillShape(X,Y: Integer);
           External Hires[10F6];
[KAK] 23: Procedure Pattern(Var P);
           External Hires[11CC];
[10N] 24: Procedure FillPattern(X1,Y1,X2,Y2: Integer);
           External Hires[1B55];
[05N] 25: Procedure GraphWindow(X1,Y1,X2,Y2: Integer);
           External Hires[1098];
[HRN] 26: Procedure FillWindow(Bitmuster: Integer);
           External Hires[10B4];
[CKW] 27: Procedure InvertWindow;
           External Hires[10AF];
[53N] 28: Procedure CopyWindow(Seite,X,Y: Integer);
           External Hires[1107];
[R4L] 29: Function  GraphMode: Boolean;
           External Hires[10C6];
[W2P] 30: Function  GetDotColor(X,Y: Integer): Integer;
           External Hires[1139];
[HXM] 31: Function  WhereXGraphic: Integer;
           External Hires[117A];
[UXM] 32: Function  WhereYGraphic: Integer;
           External Hires[118B];
```



```

[W1H] 33: Procedure GotoXYText(X,Y:Integer); Begin Go
toXY(X,Y); End;
[7VH] 34: Function WhereXText:Integer; Begin WhereXT
ext:=WhereX End;
[FVH] 35: Function WhereYText:Integer; Begin WhereYT
ext:=WhereY End;
[M1W] 36: Procedure GotoXY(X,Y:Integer); Begin If Gra
phMode Then GotoXYGraphic(X,Y) Else GotoXYTe
xt(X,Y) End;
[SAO] 37: Function WhereX:Integer; Begin If GraphMod
e Then WhereX:=WhereXGraphic Else WhereX:=Wh
ereXText End;
[YAO] 38: Function WhereY:Integer; Begin If GraphMod
e Then WhereY:=WhereYGraphic Else WhereY:=Wh
ereYText End;
[NVO] 39: Procedure LeaveGraphic; Begin If GraphMode
Then Begin TextMode; ClrScr; ConOutPtr:=ConO
utPtrText End End;
[14X] 40: Procedure InitGraphic; Begin If Not GraphMo
de Then Begin Hires; GraphWindow(0,0,XMaxGlb
,YMaxGlb);
[67R] 41: SelectScreen(1); GotoXY(1,1); SetCursor(7,8
); HiresColor(White); SetLineStyle(-1);
[PHR] 42: ClearScreen; ConOutPtrText:=ConOutPtr; ConO
utPtr:=Ofs(WriteCharacter) End End;

```

Listing 6. Der Quellcode für die Pascal-Definitionen »hi-res.pas«

```

[3NA] 0: {$I Hires Grafikroutinen laden. }
[000] 1:
[VL1] 2: Const
[3SD] 3: Step=2; { Schrittweite für die bewegte
Grafik. }
[EFE] 4: FuellMuster: Array[1..8] Of Byte = (129
,66,36,24,24,36,66,129);
[000] 5:
[ELO] 6: Var
[8T4] 7: X,Y,XR,YR: Integer;
[H83] 8: E: Char;
[000] 9:
[BG1] 10: Begin
[NH3] 11: InitGraphic;
[TGB] 12: SetCursor(0,0); { Cursor abschalten. }
[SV9] 13: DrawText(88,0,4,5,'Bewegte Grafik:');
[YGL] 14: SelectScreen(2); { Bild in der 2. Bildsch
irmseite vorbereiten. }
[MG3] 15: ClearScreen;
[CS3] 16: Circle(50,40,40);
[6S3] 17: Circle(50,40,30);
[H34] 18: FillShape(12,40);
[OS6] 19: DrawString(30,36,'Test!');
[PN5] 20: GraphWindow(0,0,100,80);
[7AF] 21: X:=100; Y:=100; XR:=1; YR:=1; { Variablen
initialisieren. }
[X42] 22: Repeat
[9WE] 23: CopyWindow(1,X,Y); { Fensterinhalt vers
chieben. }
[L83] 24: X:=X+Step*XR;
[V83] 25: Y:=Y+Step*YR;
[2PL] 26: If (X>XMaxGlb-120) Or (X<10) Then XR:=-
XR; { Bewegungsrichtung umkehren. }
[A79] 27: If (Y>YMaxGlb-100) Or (Y<60) Then YR:=-
YR;
[CA5] 28: Until Keypressed;
[4T2] 29: Read(Kbd,E);
[HG4] 30: SelectScreen(1);
[6R9] 31: GraphWindow(0,0,XMaxGlb,YMaxGlb);
[EZH] 32: Pattern(Fuellmuster); { Bildschirm mit Mu
ster füllen. }
[1R9] 33: FillPattern(0,0,XMaxGlb,YMaxGlb);
[A1S] 34: GraphWindow(150,100,570,250); { Neues, kl
einere Fenster definieren und löschen. }
[VZ3] 35: FillWindow(0);
[ZKC] 36: DrawText(10,2,2,2,'Ausgabe in einem Fenst
er');
[GLG] 37: GraphWindow(155,120,565,245); { Ausgabefe
nster definieren. }
[MNG] 38: For X:=1 To 200 Do Write('Testausgabe ')
; { Text ausgeben. }
[2V5] 39: Writeln; Writeln;
[4FD] 40: SetCursor(7,8); { Strichcursor erzeugen.
}
[BNA] 41: Write(' Beliebige Taste drücken!');
[4T2] 42: Read(Kbd,E);
[TP3] 43: LeaveGraphic
[5SO] 44: End.

```

Listing 7. Pascal-Listing zur Demonstration der Grafik-Routinen »demo.pas«

C-TOOLS

C-terp der prof. C-Interpreter

neu: für TURBO C nur.....DM 350,-

DeSmet C Development Package

inkl. Debugger.....DM 295,-

GraphiC wiss. Präsentationsgrafik

jetzt neue Version 4.1.....DM 855,-

Ip C-Druckerfunktionen

für jeden Drucker.....DM 225,-

Vance C-lib Window Bibliothek

(UNIX "curses" kompatibel).....DM 295,-

EGA-Tools im C und Assembler Source

von COBRA.....DM 195,-

MID Treiber für beliebig viele

V24-Schnittstellen.....DM 285,-

PC-lint C Syntax- und Semantik-

prüfung über mehrere Files.....DM 399,-

VTEK Tektronix 4014 und VT 100

Terminal-Emulation.....DM 350,-

PrintQ der Druckerspooler für den PC.....DM 285,-

BTREE + ISAM neue Version 2.6 **DM 350.-**

jetzt mit: File Recovery Utility, alle Datentypen als Index

ab sofort: spezielle Version für MS-QuickBasic 4.0 DM 245.- !!!

Viele Produkte mehr: Info anfordern! Keine zusätzlichen Versandkosten!

KESSLER Softwareentwicklung Mitteldorfstr.17 3400 Göttingen Tel. 0551-792488

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, über das 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 das Schreiben anfangen: 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 reines ASCII-Feld 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.

Redaktion

DOS International

Hans-Stießberger-Str. 3

8013 Haar/München

Tel.: (089) 46 10 01-0

Grafikfunktionen für Turbo C

Wer sich Turbo C vor Erscheinen der Version 1.5 gekauft hat, wird erschreckt festgestellt haben, daß im Funktionsumfang keinerlei Grafikroutinen implementiert wurden. Aus diesem Grund sorgt die folgende Sammlung von Grafikfunktionen für Abhilfe.

Damit auch Besitzer einer älteren Version von Turbo C in den Genuß von Grafikfunktionen kommen, ergänzt das Programm »video.c« (Listing 1) den Funktionsumfang des C-Compilers um 15 Grafikfunktionen. Das Programm »video.c« erkennt automatisch, ob in dem PC, auf dem »video.c« benutzt wird, ein Hercules- oder CGA-kompatibler Bildschirmadapter vorhanden ist, und stellt sich entsprechend darauf ein.

Die Grafikfunktionen von »video.c« sind nicht auf Programmmlänge, sondern auf Ausführungsgeschwindigkeit optimiert, weshalb der Quellcode teilweise einen etwas redundanten Eindruck macht.

Die 15 Grafikfunktionen von »video.c« sind:

void initvideo()

Diese Funktion dient zum Initialisieren der Grafikroutinen von »video.c«.

void hires()

Die Hires-Funktion schaltet den Grafikmodus ein.

void text()

Die Text-Funktion schaltet in den Textmodus zurück. Ist im PC eine Hercules-Grafikkarte vorhanden, so wird die Diag-Konfiguration gewählt, in der sich die Herculeskarte wie ein Monochromadapter verhält.

intgetcard()

Diese Funktion gibt den Wert 0 zurück, sofern der verwendete Bildschirmadapter CGA-kompatibel ist. Handelt es sich bei dem Bildschirmadapter dagegen um einen Monochromadapter oder eine Hercules-Grafikkarte, so wird der Wert 1 zurückgegeben.

void gcls(int mod)

Die Funktion »gcls« löscht (mod = 0), setzt (mod = 1) oder invertiert (mod = 2) den gesamten Grafikbildschirm. Diese Funktion sollte nur im Grafikmodus, der mit der Hires-Funktion aufgerufen wird, benutzt werden.

void dot(int x, int y, int mod)

Die Dot-Funktion setzt (mod = 1), löscht (mod = 0) oder invertiert (mod = 2) einen Grafikpunkt mit der Koordinate x/y.

void line(int x1, int y1, int x2, int y2, int mod)

Die Line-Funktion zeichnet (mod = 1), löscht (mod = 0) oder invertiert (mod = 2) eine Linie zwischen den Koordinaten x1/y1 und x2/y2.

void lineto(int x, int y, int mod)

Diese Funktion zeichnet (mod = 1), löscht (mod = 1) oder invertiert (mod = 2) eine Linie vom Endpunkt der zuletzt mit »lineto« gezeichneten Linie bis zu der Koordinate x/y. Diese Koordinate wird dann zum Startpunkt für die nächste Lineto-Funktion. Wird »lineto« zum erstenmal aufgerufen, so muß zuvor mit setxy(intx, int y) die Koordinate x/y als Startpunkt für die erste Linie gesetzt werden.

void frame(int x1, int y1, int x2, int y2, int mod)

Die Frame-Funktion zeichnet (mod = 1), löscht (mod = 0) oder invertiert (mod = 2) einen Rahmen. Der linke, obere Eckpunkt des Rahmens wird durch die Koordinate x1/y1 und der rechte, untere Eckpunkt durch die Koordinate x2/y2 definiert.

void setwindow(int x1, int y1, int x2, int y2, int mod)

Diese Funktion legt die Größe eines Grafikfensters fest. Der linke, obere Eckpunkt des Fensters wird durch die Koordinate x1/y1 und der rechte, untere Eckpunkt durch die Koordinate x2/y2 definiert. Wenn mod auf 1 gesetzt ist, wird ein Rahmen um das Grafikfenster gezeichnet und das Fenster exakt um die Rahmengröße verkleinert. Alle Grafikbefehle werden im folgenden nur noch innerhalb des Grafikfensters beziehungsweise innerhalb des Rahmens ausgeführt. Ist kein Grafikfenster definiert, so wird der gesamte Bildschirm als Fenster angesehen, dessen Größe natürlich von Grafikkarte zu Grafikkarte verschieden ist.

void gwindow(int mod)

Diese Funktion setzt (mod = 0), löscht (mod = 1) oder invertiert (mod = 2) das mit »setwindow« definierte Grafikfenster.

gotoxy(intx, int y);

Die Gotoxy-Funktion plaziert den Cursor im Textmodus auf der durch x und y definierten Textkoordinate.

fontsize(int groesse)

Diese Funktion legt im Grafikmodus die Schriftgröße fest.

gprint(int x, int y, string, int mod, int spezial)

Diese Funktion gibt im Grafikmodus eine Zeichenkette (String) ab der Koordinate x,y mit der durch »fontsize« festgelegten Größe aus. Die Zeichenkette darf nur Zeichen mit ASCII-Werten zwischen 32 und 125 enthalten.

Die Koordinate x/y definiert die linke obere Ecke des ersten Buchstabens der Zeichenkette. Die Bildpunkte der Zeichen werden wahlweise gesetzt (mod = 0), gelöscht (mod = 1) oder invertiert (mod = 2).

Die Zeichenkette läßt sich von links nach rechts (spezial = 0), von oben nach unten bei Drehung der Buchstabenmatrizen um 90 Grad im Uhrzeigersinn (spezial = 1) oder von unten nach oben bei Drehung der Buchstabenmatrizen um 90 Grad gegen den Uhrzeigersinn (spezial = 2) ausgeben. Auf diese Weise lassen sich Grafiken vielfältig beschriften.

Bei der Hercules-Grafikkarte behilft sich die Funktion »gprint« mit einem kleinen Trick

Die Funktion der Gprint-Funktion ist allgemein recht trickreich, denn während die CGA-Farbgrafikkarte im Grafikmodus Text ausgeben kann, ist dies bei der Hercules-Grafikkarte zunächst einmal nicht der Fall.

Im BIOS eines PC ist eine Tabelle der Bildpunktmatrizen für die ASCII-Zeichen 0 bis 127 abgelegt. Die Größe der Bildpunktmatrix für ein Zeichen beträgt in der Tabelle 8x8 Bildpunkte. Diese Tabelle benutzt das Betriebssystem bei der CGA-Farbgrafikkarte für die Darstellung von Zeichen auf dem Bildschirm.

Bei der Hercules-Grafikkarte ist dagegen keine Tabelle mit den erforderlichen 9x14-Bildpunktmatrizen für die ASCII-Zeichen vorgesehen. Natürlich läßt sich für die Herculeskarte eine solche Tabelle selbst anlegen, doch dann müßten Sie genau 3584 Werte aus einem Listing abtippen. Damit Sie sich diese Arbeit sparen können, wurde das Schriftproblem für die Hercules-Grafikkarte gelöst, indem von »video.c« die Bildpunktmatrizen für die CGA-Karte eingelesen und entsprechend dem Wert für »fontsize« vergrößert werden.

Daß ein vergrößerter Buchstabe mit einer 8x8-Matrix natürlich nicht so gut aussieht, wie ein Buchstabe mit einer 9x14-Matrix, ist klar. Dadurch, daß sich die Buchstaben mit »gprint« an einer beliebigen Position auf dem Bildschirm darstellen und auch drehen lassen, ist die Gprint-Funktion zudem noch relativ langsam. Alles in allem ist diese Funktion jedoch sehr vielseitig und in vielen Situationen gut zu gebrauchen.

Die durch die Hardware vorgegebenen Wertebereiche für die Grafikkoordinaten reichen bei der CGA-Farbgrafikkarte für x von 0 bis 639 und für y von 0 bis 199. Bei der Hercules-Grafikkarte gilt für x ein Wertebereich von 0 bis 719 und für y ein Wertebereich von 0 bis 347. Sicherheitsabfragen verhindern, daß außerhalb des Bildschirmspeichers Daten überschrieben werden können.

Das Programm »demo.c« (Listing 2) führt anhand einiger Beispiele die Fähigkeiten der beschriebenen Grafikfunk-

tionen vor. Um aus »video.c« und »demo.c« ein lauffähiges Demonstrationsprogramm zu erzeugen, müssen Sie eine Projektdatei, beispielsweise mit dem Dateinamen »grafik.prj«, anlegen und in diese Datei die beiden Zeilen

```
video.c
demo.c
```

schreiben. Wenn Sie jetzt in Turbo C als Projekt den Dateinamen »grafik.prj« angeben und darauf die Compilierung in eine EXE-Datei starten, übersetzt der Compiler »video.c« und »demo.c« in Objektdateien und linkt diese anschließend zu dem Programm »grafik.exe« zusammen. Das Grafik-Programm können Sie darauf ganz einfach durch Eingabe von

```
grafik
```

aus der MS-DOS-Kommandoebene starten.

(Bernhard Brandt/ma)

```
[A4V] 0: /* Mit video.c bedient Turbo-C gleichzeitig
      die Grundgrafik der CGA- und Hercules-Karte
      */
[9GC] 1: /* Programmiert von Bernhard Brandt (1988)
      */
[000] 2:
[625] 3: # include <dos.h>
[XKK] 4: # define OFFTA offstemp=(!card) ? (y1 >> 1
      << 4) : ((y1 >> 2 << 1) + (y1 >> 2 << 4));
[LEM] 5: # define OFFTB harray[y1 & (card << 1 | 1)
      ] + offstemp + (offstemp << 2)+(x1>>3)
[MGH] 6: /* offta, offtb: offset zum videosegment b
      erechnen */
[5T2] 7: # define SETPOINT { switch (mod) { case 0:
      *s &= ~ bitpos; break; case 1: *s |= bitpos;
      break; case 2: *s ^= bitpos; break; } }
[XOD] 8: /* Punkt löschen, setzen, invertieren */
[1TM] 9: # define VERT panzahl--; (unsigned int) s =
      temp+fanzahl; SETPOINT y1++
[8FP] 10: # define DECRX { dd=ddi; x1--; bitpos <=
      1; if (!bitpos) { bitpos++; fanzahl--; } }
[7RG] 11: # define TEST ((y1>ymax) && (y1<=ymax) &&
      (x1<=xmax) && (x1>=xmin))
[FSL] 12: # define TESTANDDOT if TEST { (unsigned int)
      s = temp+fanzahl; SETPOINT}
[87B] 13: static unsigned char bitpos,framflg;
[YSC] 14: /* bitpos bitmaske für byte im videoram */
[TMC] 15: static unsigned int segment,size,siss;
[6XA] 16: /* segment = segment des videorams */
[YET] 17: static int xmin,xmax,ymin,ymax,glxmin,glxma
      x,glymin,glymax,card,glxold,glyold;
[HKG] 18: /* xmin,x.. Minimax-Koordinaten des Grafikf
      ensters */
[H8K] 19: /* glxmin,gl..Hardware-Minimax-Koordinaten
      des Bildschirms */
[2NL] 20: static unsigned char helparray[8] = {128,64,
      32,16,8,4,2,1}; /* für bitpos */
[99M] 21: static unsigned int harray[4] = {0,0x2000,0
      x4000,0x6000}; /* für HGA yinkrem */
[7CF] 22: static unsigned char byt[128][8]; /* Bios S
      chrift 8*8 */
[000] 23:
[5V5] 24: void mode(int value)
[V30] 25: {
[905] 26: union REGS regs;
[7W4] 27: regs.x.ax = value;
[V86] 28: int86(0x10,&regs,&regs);
[X30] 29: }
[YS5] 30: void initvideo(void)
[V30] 31: {
[SD2] 32: char bb;
[367] 33: unsigned int offs,asc;
[8L1] 34: int a;
[KS6] 35: for (asc=0;asc<128;asc++)
[V30] 36: {
[YE5] 37: offs = 0x0a6d + (asc<<3);
[EWC] 38: for (a=0;a<8;a++) byt[asc][a] = peekb(0xf
      f00,++offs);
[X30] 39: }
[8YN] 40: card=xmin=glxmin=0; /* Minimum Bildschir
      mkoordinaten */ /* default CGA */
[454] 41: ymin=glymin=0;
[LWB] 42: segment=0xb000; /* default HGA segment */
[SKN] 43: bb=peekb(0x0000,0x0449); /* vom BIOS verw
      endete Speicherstelle auslesen */
[VR6] 44: if (bb<7) /* cga modi */
```

```
[V30] 45: {
[XCB] 46: segment+=0x0800; /* segment = segment CG
      A */
[V7D] 47: xmax = glxmax= 639; /* Bildschirmfenster
      auf CGA */
[FB4] 48: ymax = glymax= 199;
[X30] 49: }
[A54] 50: else if (bb==7)
[V30] 51: {
[3N8] 52: card++; /* card = Hercules */
[SDD] 53: xmax=glxmax=719; /* Bildschirmfenster auf
      HGA */
[8H4] 54: ymax=glymax=347;
[X30] 55: }
[A11] 56: else
[V30] 57: {
[CKF] 58: printf("Kein MDA-, HGA- oder CGA-Adapter i
      nstalliert!");
[NR3] 59: while(!getch());
[X30] 60: }
[X30] 61: }
[FD5] 62: int getcard(void)
[V30] 63: {
[E37] 64: return(card); /* 0 CGA, 1 HGC */
[X30] 65: }
[OK9] 66: void dot(int x1, int y1, int mod)
[V30] 67: {
[2H8] 68: unsigned int offstemp,merk;
[186] 69: unsigned char _es *s;
[2E2] 70: merk=_ES;
[W32] 71: if TEST
[V30] 72: {
[DE3] 73: _ES=segment;
[221] 74: OFFTA
[HS6] 75: (unsigned int) s = OFFTB;
[NG7] 76: bitpos = helparray[x1 & 7];
[W32] 77: SETPOINT
[X30] 78: }
[EN2] 79: _ES=merk;
[X30] 80: }
[D1H] 81: void gotoxy(unsigned char spalte,unsigned c
      har zeile)
[V30] 82: {
[905] 83: union REGS regs;
[4M3] 84: regs.h.ah = 15;
[XMM] 85: int86(0x10,&regs,&regs); /* Videomodus er
      mitteln und in BX kopieren */
[H05] 86: regs.h.dh=zeile;
[855] 87: regs.h.dl=spalte;
[TT2] 88: regs.h.ah=2;
[MLG] 89: int86(0x10,&regs,&regs); /* normale Routin
      e für gotoxy */
[X30] 90: }
[925] 91: void gcls(int mod)
[V30] 92: {
[6M5] 93: unsigned int merk;
[ZE7] 94: unsigned char _es *chp;
[LH6] 95: merk=_ES; _ES=segment;
[Y8B] 96: (unsigned int) chp=(card) ? 0x8000 : 0x400
      0;
[7SG] 97: if (card) outportb(0x03b8,2); /* Grafik + B
      ildschirm aus */
[EC6] 98: else outportb(0x03d8,16);
[NW2] 99: switch(mod)
[V30] 100: {
[WBA] 101: case 0: while(chp) { *--chp=0; } break;
```



```

[SVA] 102: case 1: while(chp) { *--chp=0xff; } brea
k;
[DRA] 103: case 2: while(chp) { *--chp^=0xff; } bre
ak;
[X30] 104: }
[Y28] 105: if (card) outportb(0x03b8,10);
[6C6] 106: else outportb(0x03d8,24);
[EN2] 107: _ES=merk;
[X30] 108: }
[000] 109:
[BD7] 110: static void hercgrafik()
[V30] 111: {
[B75] 112: unsigned char a;
[W8C] 113: unsigned char gtable[12]={0x36,0x2d,0x2e,0
x07,
[3A4] 114: 0x5b,0x02,0x57,0
x57,
[YL4] 115: 0x02,0x03,0x00,0
x00};
[L1D] 116: outportb(0x03bf,3); /* alle Modi HGA mögli
ch */
[G8D] 117: outportb(0x03b8,2); /* Grafik + Bildschirm
aus */
[B0C] 118: for (a=0;a<12;a++) /* Grafik initialisier
en */
[V30] 119: {
[3A5] 120: outportb(0x03b4,a);
[R18] 121: outportb(0x03b5,gtable[a]);
[X30] 122: }
[7G9] 123: gcls(0); /* Videoram 0 setze
n */
[LMB] 124: outportb(0x03b8,10); /* Bildschirm ein */
[X30] 125: }
[000] 126:
[AX6] 127: static void cgagrafik()
[V30] 128: {
[RX3] 129: mode(6); gcls(0);
[X30] 130: }
[7R2] 131: /* htext()
[V30] 132: {
[8L1] 133: int a;
[KEC] 134: unsigned char ttable[12]={0x61,0x50,0x52,0
x0f,
[D44] 135: 0x19,0x06,0x19,0
x19,
[M35] 136: 0x02,0x0d,0x0b,0
x0c};
[TF5] 137: outportb(0x03b8,32);
[8Y3] 138: for (a=0;a<12;a++)
[V30] 139: {
[3A5] 140: outportb(0x03b4,a);
[428] 141: outportb(0x03b5,ttable[a]);
[X30] 142: }
[PF5] 143: outportb(0x03b8,40);
[GN0] 144: } /*
[31M] 145: /* Die Routine htext() wird beim Aufruf vo
n text() nicht benutzt, */
[T2H] 146: /* denn mit outportb(0x03bf,1); mode(7); w
ird der MDA-Modus der */
[L2M] 147: /* Herculeskarte oder auch die Diag-Konfig
uration eingestellt */
[6SM] 148: /* Möchten Sie den Textmodus der Herculesk
arte aktivieren, dann */
[FOA] 149: /* verwenden sie diese Routine */
[000] 150:
[000] 151:
[N33] 152: void hires()
[V30] 153: {
[Y76] 154: if (!card) cgagrafik();
[8G5] 155: else hercgrafik();
[X30] 156: }
[TV2] 157: void text()
[V30] 158: {
[SS4] 159: if (!card) mode(2);
[A11] 160: else
[V30] 161: {
[538] 162: outportb(0x03bf,1); mode(7);
[X30] 163: }
[X30] 164: }
[000] 165:
[TBC] 166: line (int x1, int y1, int x2, int y2, int
mod)
[433] 167: /* Linien */
[4CL] 168: /* schnelle, aber längere Routine, da die v
orkommenden Fälle */
[UAH] 169: /* CGA, HGA, vertikal, horizontal usw. in e
igenen Schleifen */
[E66] 170: /* behandelt werden */
[V30] 171: {
[TH6] 172: unsigned char _es *s;
[TCG] 173: unsigned int offstemp,temp,offseta,fanzahl
,merk;
[5A9] 174: int panzahl,dsi,dd,tempb,ddi;
[7E5] 175: unsigned char xr;
[2E2] 176: merk=_ES;

```

```

[DE3] 177: _ES=segment;
[YUA] 178: if (y1 == y2) /* horizontal
Beginn */
[LND] 179: { /* diese Unte
rroutine ermittelt */
[UZE] 180: if ((y1==ymin) && (y1<=ymax)) /* die Anza
hl voller Bytes, */
[20D] 181: { /* die somi
t beim Ziehen einer */
[X2C] 182: x2 = (x2>xmax) ? xmax : ((x2<xmin) ? xmi
n : x2);
[H2C] 183: x1 = (x1>xmax) ? xmax : ((x1<xmin) ? xmi
n : x1);
[RUD] 184: if (x1>x2) /* horizontale
n Linie direkt, */
[A7E] 185: { /* acht Bit au
f einmal, manipuliert */
[VUA] 186: dd=x1; x1=x2; x2=dd; /* werden könn
en */
[X30] 187: }
[KWF] 188: panzahl = (x2-x1); /* formale Anza
hl volle Bytes */
[97F] 189: fanzahl = panzahl >> 3; /* Punktzahl
durch Acht */
[E23] 190: panzahl++;
[221] 191: OFFTA
[HS6] 192: (unsigned int) s = OFFTB;
[059] 193: bitpos = helparray[xr = x1 & 7];
[1F4] 194: while (panzahl)
[V30] 195: {
[G23] 196: panzahl--;
[W32] 197: SETPOINT
[M6A] 198: if (xr!=7) bitpos = helparray[++xr];
[A11] 199: else
[V30] 200: {
[PH0] 201: s++;
[MMG] 202: if (fanzahl>2) /* tatsächliche Anz
ahl voller Bytes im */
[3KD] 203: { /* Videoram kann fo
rmale Anzahl - 2 sein, */
[DVE] 204: fanzahl--; /* da ja links und
rechts, je nach */
[V4L] 205: bitpos=255; /* Koordinaten, tei
lgefüllte Bytes auftauchen */
[Z2B] 206: panzahl-=7; /* z.B. 0111 1111 B
ytes 1111 1110 */
[X30] 207: }
[A11] 208: else
[V30] 209: {
[U20] 210: xr=0;
[B13] 211: bitpos=128;
[LU1] 212: }}}}
[UA8] 213: else if (x1 == x2) /* vertikal */
[V30] 214: {
[EK6] 215: if ((x1==xmax)&&(x1>=xmin))
[V30] 216: {
[E21] 217: if (y1>y2)
[V30] 218: {
[PH4] 219: dd=y1; y1=y2; y2=dd;
[X30] 220: }
[X3C] 221: y2 = (y2>ymax) ? ymax : ((y2<ymin) ? ymi
n : y2);
[H3C] 222: y1 = (y1>ymax) ? ymax : ((y1<ymin) ? ymi
n : y1);
[W35] 223: panzahl = (y2-y1);
[E23] 224: panzahl++;
[221] 225: OFFTA
[UT8] 226: temp = harray[y1 & (card << 1 | 1)];
[LCC] 227: fanzahl = offstemp + (offstemp << 2)+(x1
>>3);
[NG7] 228: bitpos = helparray[x1 & 7];
[DD2] 229: if (!card)
[V30] 230: {
[2B4] 231: while(panzahl)
[V30] 232: {
[D21] 233: VERT;
[WLM] 234: temp ^= 0x2000; /* wenn y gerade, dan
n jetzt ungerade und umgekehrt */
[1M5] 235: if (!(y1 & 1)) fanzahl+=80;
[X30] 236: }
[X30] 237: }
[A11] 238: else
[V30] 239: {
[1F4] 240: while (panzahl)
[V30] 241: {
[D21] 242: VERT;
[FPH] 243: temp += (temp==0x6000) ? 40960 : 0x200
0; /*im ersten Fall Überlauf auf */
[R1F] 244: if (!temp) fanzahl+=90;
/* Null = 65536 erzeugen */
[271] 245: }}}}
[M1H] 246: else /*andere Linien weder horizo
ntal noch vertikal */
[V30] 247: {
[MD9] 248: if (x2>x1) /* Startpunkte ordnen */

```



```

[V30] 249: {
[U9] 250:   dd=x1; x1=x2; x2=dd; dd=y1; y1=y2; y2=dd
;
[X30] 251: }
[MX5] 252: dsi = panzahl = (x1-x2);
[UC7] 253: panzahl+= ddi = abs (y2-y1);
[KN8] 254: dd=(dd>dsi) ? -(ddi>>1) : (dsi>>1);
[E23] 255: panzahl++;
[221] 256: OFFTA
[UT8] 257: temp = harrayb[y1 & (card << 1 | 1)];
[LCC] 258: fanzahl = offstemp + (offstemp << 2)+(x1
>3);
[NG7] 259: bitpos = helparray[x1 & 7];
[CF5] 260: if (!card) /* CGA */
[V30] 261: {
[7W7] 262:   if (y2>y1) /* erster Fall
1 */
[V30] 263:   {
[1F4] 264:     while (panzahl)
[V30] 265:     {
[G23] 266:       panzahl--;
[L82] 267:       TESTANDDOT
[752] 268:       if (dd<=0)
[V30] 269:       {
[OX3] 270:         dd+=dsi; /* y*/
[7R3] 271:         temp ^= 0x2000;
[A16] 272:         if (!(++y1 & 1)) fanzahl+=80;
[X30] 273:       }
[7A2] 274:       else DECRX
[X30] 275:       }
[X30] 276:     }
[Al1] 277:   else
[V30] 278:   {
[CA7] 279:     while (panzahl) /*y2 < y1 */
[V30] 280:     {
[G23] 281:       panzahl--;
[L82] 282:       TESTANDDOT
[752] 283:       if (dd<=0)
[V30] 284:       {
[T14] 285:         dd+=dsi; /* y */
[7R3] 286:         temp ^= 0x2000;
[UR5] 287:         if (--y1 & 1) fanzahl-=80;
[X30] 288:       }
[7A2] 289:       else DECRX
[ER0] 290:     }
[LV3] 291:   else /* HGA */
[V30] 292:   {
[9H7] 293:     if (y2>y1) /* erster Fall */
[V30] 294:     {
[1F4] 295:       while (panzahl)
[V30] 296:       {
[G23] 297:         panzahl--;
[L82] 298:         TESTANDDOT
[752] 299:         if (dd<=0)
[V30] 300:         {
[GC7] 301:           dd+=dsi; /* inkrement y*/
[FT0] 302:           y1++;
[FPH] 303:           temp += (temp==0x6000) ? 40960 : 0x20
00; /*im ersten Fall Überlauf auf */
[R1F] 304:           if (!temp) fanzahl+=90;
/* Null = 65536 erzeugen */
[X30] 305:         }
[NA2] 306:         else DECRX
[X30] 307:         }
[X30] 308:       }
[Al1] 309:     else
[V30] 310:     {
[3G7] 311:       while (panzahl) /* y2<y1 HGA */
[V30] 312:       {
[G23] 313:         panzahl--;
[L82] 314:         TESTANDDOT
[752] 315:         if (dd<=0)
[V30] 316:         {
[OX3] 317:           dd+=dsi; /* y*/
[X30] 318:           y1--;
[LB6] 319:           if (!temp) fanzahl+=90;
[LPG] 320:           temp -= (!temp) ? 40960 : 0x2000; /*i
m ersten Fall Unterlauf auf */
[S16] 321:           6000h = 24576 erzeugen */
[X30] 322:         }
[7A2] 323:         else DECRX
[271] 324:       }
[EN2] 325:       _ES=merk;
[X30] 326:     }
[X09] 327:     lineto (int x1, int y1, int mod)
[V30] 328:     {
[N19] 329:       line(glxold, glyold, x1, y1, mod);
[MA6] 330:       glxold=x1; glyold=y1;
[X30] 331:     }
[WPA] 332:     frame (int x1, int y1, int x2, int y2, int
mod)
[V30] 333:     {
[43B] 334:       line(x1,y1,x2,y1,mod); line(x2,y1,x2,y2,mo
d);

```

```

[S2B] 335:   line(x2,y2,x1,y2,mod); line(x1,y2,x1,y1,mo
d);
[X30] 336: }
[RVD] 337: setwindow (int x1, int y1, int x2, int y2,i
nt mod)
[V30] 338: {
[KCM] 339:   xmin=(x1>glxmin) ? x1: glxmin; /* wenn du
rch Hardware vorgegebene */
[8EP] 340:   xmax=(x2>glxmax) ? x2: glxmax; /* Grenzen
des Videorams überschritten */
[W6P] 341:   ymin=(y1>glymin) ? y1: glymin; /* werden
sollen, solches verhindern */
[P2E] 342:   ymax=(y2>glymax) ? y2: glymax; /* Sicherh
eit ! */
[3BP] 343:   if (framflg=mod) /* Rahmen zeichnen und du
rch Verkleinern des Fensters*/
[E8E] 344:   { /* schützen, Compilerwarnu
ng ungültig */
[8Y8] 345:     frame(xmin,ymin,xmax,ymax,1);
[UT7] 346:     xmin++; xmax--; ymin++; ymax--;
[X30] 347:   }
[X30] 348: }
[FN5] 349: gwindow(int mod)
[V30] 350: {
[OR1] 351:   int y;
[43E] 352:   for (y=ymin;y<=ymax;y++) line(xmin,y,xmax,
y,mod);
[X30] 353: }
[000] 354:
[CR5] 355: void fontsize(int a)
[V30] 356: {
[WN5] 357:   siss=size = a; siss++;
[X30] 358: }
[000] 359:
[3PG] 360: gprint(int x1,int y,char string[],int mod,i
nt spezial)
[V30] 361: {
[KR5] 362:   /* char string[]; */
[02K] 363:   unsigned char fontelement[8],kit[8],help[8
]= {1,2,4,8,16,32,64,128};
[UX7] 364:   unsigned int offs,1,asc;
[1L4] 365:   int x,a,b,c,yr;
[ZK0] 366:   i=0;
[FA6] 367:   while (string[i] != '\0')
[V30] 368:   {
[XM4] 369:     asc=string[i];
[8N0] 370:     ++i;
[7T3] 371:     for (a=0;a<8;a++)
[V30] 372:     {
[DKB] 373:       kit[a]=byt[asc][a]; fontelement[a]=0;
[X30] 374:     }
[W45] 375:     switch (spezial)
[V30] 376:     {
[EKG] 377:       case 1: /* Matrix um 90 Grad gegen U
hrzeigersinn drehen */
[LH5] 378:         for (b=0,c=7;b<8;b++,c--)
[V30] 379:         {
[7T3] 380:           for (a=0;a<8;a++)
[V30] 381:           {
[9MK] 382:             if (helparray[b] & kit[a]) fontelement
[c]^=helparray[a];
[X30] 383:           }
[X30] 384:         }
[KV1] 385:         break;
[SKF] 386:       case 2: /* Matrix um 90 Grad im Uhrzeig
ersinn drehen */
[9G5] 387:         for (b=0,c=0;b<8;b++,c++)
[V30] 388:         {
[7T3] 389:           for (a=0;a<8;a++)
[V30] 390:           {
[OBG] 391:             if (helparray[b] & kit[a]) fontelement
[c]^=help[a];
[X30] 392:           }
[X30] 393:         }
[KV1] 394:         break;
[MGG] 395:       case 0: for (a=0; a<8;a++) fontelement[a
]=kit[a]; break;
[S45] 396:     } /* end of switch */
[292] 397:     if (!size)
[V30] 398:     {
[ZCG] 399:       for (a=0,yr=y;a<8;a++,y++) /* als
Bildpunkte ausgeben */
[V30] 400:       {
[416] 401:         for (x=x1,b=0;b<8;b++,x++)
[V30] 402:         {
[8NE] 403:           if (fontelement[a] & helparray[b]) dot
(x,y,mod);
[X30] 404:         }
[X30] 405:       }
[X30] 406:     }
[Al1] 407:   else
[V30] 408:   {
[F8H] 409:     for (a=0,yr=y;a<8;a++,y+=size) /*
als Bildpunkte ausgeben */
[V30] 410:     {

```



```
[LN8] 411: for (x=x1,b=0;b<8;b++,x+=size,x++)
[V30] 412: {
[5EA] 413:     if (fontelement[a] & helparray[b])
[V30] 414:     {
[PVC] 415:         for (c=y;c1=y+size;c++) line(x,c,x+size,c,mod);
[X30] 416:     }
[X30] 417: }
[X30] 418: }
[X30] 419: }
[673] 420: if(!spezial)
[V30] 421: {
[BL4] 422:     x1+=siss<<3; y=yr;
[X30] 423: }
[YP9] 424: else if (spezial==1) y--((y-yr)<<1);
[X30] 425: }
[X30] 426: }
[A97] 427: void setxy(int x, int y)
[ZP7] 428: { glxold=x; glyold=y; }
```

Listing 1. Das Programm »video.c« enthält den Quellcode der Grafikroutinen für Turbo C

```
[SU5] 0: # include <stdio.h>
[6L5] 1: # include <math.h>
[625] 2: # include <dos.h>
[000] 3:
[OF1] 4: main()
[V30] 5: {
[R15] 6:     unsigned int a;
[8KD] 7:     int b,d,f,te,s,e,l,zz,k,fz,x,y,tbax[102],taby[102];
[Z55] 8:     double phi,step;
[PF3] 9:     initvideo();
[HY1] 10:    text();
[9C3] 11:    gotoxy(20,1);
[REK] 12:    printf("Mit Tastendruck im Demoprogramm fortsetzen");
[NR3] 13:    while(!getch());
[K32] 14:    hires();
[4U1] 15:    gcls(0);
[UK3] 16:    fontsize(1);
[LX4] 17:    gprint(10,10,"H",1,0);
[K65] 18:    gprint(26,12,"o",1,0);
[D65] 19:    gprint(42,10,"p",1,0);
[V65] 20:    gprint(58,10,"p",1,2);
[UF5] 21:    gprint(74,12,"la",1,0);
[NR3] 22:    while(!getch());
[YU4] 23:    for (fz=0;fz<6;fz++)
[V30] 24:    {
[P44] 25:        fontsize(fz);
[W62] 26:        if (fz<3)
[V30] 27:        {
[GA5] 28:            for(l=10;l<124;l+=fz+1)
[V30] 29:            {
[GL5] 30:                gprint(10,l,"D E M O",2,2);
[GL5] 31:                gprint(10,l,"D E M O",2,2);
[X30] 32:            }
[GK5] 33:            for (l=10;l<500;l+=fz+5)
[V30] 34:            {
[NS5] 35:                gprint(l,123,"D E M O",2,0);
[NS5] 36:                gprint(l,123,"D E M O",2,0);
[X30] 37:            }
[AM5] 38:            for (l=123;l>10;l-=fz+1)
[V30] 39:            {
[BZ5] 40:                gprint(500,l,"D E M O",2,1);
[BZ5] 41:                gprint(500,l,"D E M O",2,1);
[X30] 42:            }
[X30] 43:        }
[8E9] 44:        gprint(300-fz*25,10+6*fz*fz,"D e m o",1,0);
[X30] 45:    }
[NR3] 46:    while(!getch());
[4U1] 47:    gcls(0);
[GU3] 48:    fontsize(20);
[S15] 49:    gprint(10,100,"M",1,0);
[SK3] 50:    fontsize(0);
[P6D] 51:    gprint(10,10,"Das Loch im M steht so im Bild.");
[ABK] 52:    gprint(10,18,"Besser, Sie legen sich ein eigenes Zeichen an!",1,0);
[NR3] 53:    while(!getch());
[GU1] 54:    gcls(2);
[NR3] 55:    while(!getch());
[HY1] 56:    text();
[DN3] 57:    gotoxy(20,10);
[XCF] 58:    printf("49000 Punkte berechnen, setzen und löschen.");
[NR3] 59:    while(!getch());
[K32] 60:    hires();
```

```
[4U1] 61:    gcls(0);
[K14] 62:    for (e=1;e>=0;e--)
[V30] 63:    {
[852] 64:        srand(1);
[S4B] 65:        for (a=0; a<50000;a++) dot(rand())>>5,rand()>>5,e);
[X30] 66:    }
[NR3] 67:    while(!getch());
[4U1] 68:    gcls(0);
[SK3] 69:    fontsize(0);
[ZNN] 70:    gprint(10,50,"Erst einen, dann 50 Rahmen zeichnen und wieder löschen.",1,0);
[NR3] 71:    while(!getch());
[GU1] 72:    gcls(2);
[NR3] 73:    while(!getch());
[4U1] 74:    gcls(0);
[DCB] 75:    frame(0,0,639+80*getcard(),199+148*getcard(),1);
[NR3] 76:    while(!getch());
[6CB] 77:    frame(0,0,639+80*getcard(),199+148*getcard(),0);
[K14] 78:    for (e=1;e>=0;e--)
[V30] 79:    {
[852] 80:        srand(1);
[XNG] 81:        for (a=0; a<49;a++) frame(rand())>>7,rand()>>7,rand()>>6,rand()>>7,e);
[NR3] 82:        while(!getch());
[X30] 83:    }
[HY1] 84:    text();
[N1N] 85:    printf("Präzise Linien einmal von 0/0 nach 639/1 (CGA) bzw. 719/1 (HGA) ziehen.\n");
[Z2L] 86:    printf("und einmal von 0/0 nach 1/199 (CGA) bzw. 1/347 (HGA) ziehen.\n");
[NR3] 87:    while(!getch());
[K32] 88:    hires();
[OY7] 89:    line(0,0,(getcard()) ? 719 : 639,1,1);
[7T7] 90:    line(0,0,1,(getcard()) ? 347 : 199,1);
[NR3] 91:    while(!getch());
[HY1] 92:    text();
[CC3] 93:    gotoxy(0,10);
[RSI] 94:    printf("Präzise Linien von 0/0 nach -1,199 (CGA) bzw. -1,347 (HGA) ziehen.\n");
[VNS] 95:    printf("Wo schneidet diese Gerade theoretisch den linken Bildschirmrand?\n");
[7ER] 96:    printf("Programmieren Sie diese Linie mit professionellen Toolboxes!\n");
[NR3] 97:    while(!getch());
[K32] 98:    hires();
[N58] 99:    line(0,0,-1,(getcard()) ? 347 : 199,1);
[NR3] 100:    while(!getch());
[HY1] 101:    text();
[CC3] 102:    gotoxy(0,10);
[YE3] 103:    step=0.062831853;
[8VB] 104:    for (zz=0,phi=0.0;phi<6.34;phi+=step,zz++)
[V30] 105:    {
[NHB] 106:        tbax[zz]=100+70*cos(phi)-30*cos(30*phi);
[RHB] 107:        taby[zz]=100+70*sin(phi)-30*sin(30*phi);
[X30] 108:    }
[ZL7] 109:    printf("Linien ziehen.\n");
[CZH] 110:    printf("Anschließend Tasten zum Invertieren drücken.");
[NR3] 111:    while(!getch());
[K32] 112:    hires();
[SZ7] 113:    setxy(215+tbax[0],taby[0]);
[ZOC] 114:    for (a=1;a<101;a++) lineto(215+tbax[a],taby[a],1);
[EU5] 115:    for (te=0; te<10; te++)
[V30] 116:    {
[GU1] 117:        gcls(2);
[E34] 118:        while(!getch());
[X30] 119:    }
[ZSB] 120:    gcls(0); /* Grafikbildschirm löschen */
[H55] 121:    for (f=10; f<=80; f+=10)
[V30] 122:    {
[7Y6] 123:        setwindow(f,f,110+f,110+f,1);
[BA3] 124:        gwindow(1);
[NR3] 125:        while(!getch());
[5GA] 126:        setxy(f+5+(tbax[0]>>1),f+5+(taby[0]>>1));
[ENF] 127:        for (a=1;a<101;a++) lineto(f+5+(tbax[a]>>1),f+5+(taby[a]>>1),0);
[LE5] 128:        for (te=0; te<9; te++)
[V30] 129:        {
[CA3] 130:            gwindow(2);
[NR3] 131:            while(!getch());
[X30] 132:        }
[X30] 133:    }
[NR3] 134:    while(!getch());
[HY1] 135:    text();
[59E] 136:    printf("Viel Spaß beim Ausbau von video.c.\n");
[X30] 137: }
```

Listing 2. »demo.c« führt die Fähigkeiten der Grafikroutinen eindrucksvoll vor

Ein PC mit Sound

Es herrscht landläufig die Meinung, daß Soundprogrammierung in kommerziellen Applikationen nichts zu suchen habe. Die Antwort: Im Prinzip ja, andererseits erspart eine akustische Warnung manche überflüssige Tipparbeit.

Eingangs einiges Grundlegende über die Entstehung eines Tones über den Lautsprecher Ihres PC. Ein Ton kommt durch die Schwingung der Lautsprechermembrane zustande, die wiederum die Luftpartikel zum Schwingen bringt. Geht diese Membrane nach außen, drückt sie die Luftpartikel zusammen, bewegt sie sich nach innen, entlastet sie die Luftpartikel. Geschieht diese Bewegung der Membrane nach außen und nach innen einmal in der Sekunde, so spricht man von einer Tonfrequenz von einem Hertz.

Um nun Töne mit einem Programm zu erzeugen, gibt es zwei Verfahrensweisen, den Lautsprecher des PC anzusprechen. Entweder direkt, indem man die zwei Bit des programmierbaren Peripheriegeräteprozessors (PPI, »programmable peripheral interface chip«), die den Lautsprecher bedienen, kontrolliert. Dabei muß das Programm das Timing der Impulse und der damit verbundenen Tonfrequenz überwachen. Von dieser Programmierung ist abzuraten, weil dabei die Abhängigkeit von der Arbeitsfrequenz des Computers sehr groß ist.

Bei der zweiten Methode wird mit dem Programm der Timerchip oder Soundchip (8253-Prozessor) angesprochen, um den Lautsprecher mit den gewünschten Impulsfrequenzen zu versehen. Dadurch erreichen Sie, daß bei einem PC/XT wie auf einem AT beispielsweise der Kamerton A auch wirklich die gewünschte Frequenz von 440 Hertz besitzt. Dies ist dadurch möglich, weil der 8253-Prozessor nicht nur Töne erzeugt, sondern auch noch für die Echtzeituhr des Computers verantwortlich ist.

Der 8253-Prozessor empfängt ein Signal vom Prozessor, der für die Uhr zuständig ist (8284). Der 8284 arbeitet mit einer Frequenz von 1193180 Zyklen in der Sekunde, was 1,193 MHz entspricht. Immer wenn 65536 Zyklen abgearbeitet sind, löst der Soundchip einen »clock interrupt« (Interrupt 8) aus. Dieser Interrupt ereignet sich in der Sekunde ungefähr 18 Mal (ergibt sich aus der Division von 1193180 geteilt durch 65536) und wird als »clock tick« bezeichnet. Das BIOS errechnet aus diesen »clock ticks« die Systemzeit und den Tag und sendet selbst wiederum einen eigenen Interrupt (Interrupt 28) aus.

Einige Programme machen von diesem Interrupt jedoch keinen Gebrauch, weil 18,2 Zyklen in der Sekunde für eine genaue Tonerzeugung nicht ausreichen. Basic spricht beispielsweise den Soundchip direkt an und veranlaßt ihn, den »clock tick« viermal schneller auszuführen. In diesem Fall ergibt sich dann ein Zyklus von 72,8 statt 18,2 in der Sekunde. Das BIOS wird getäuscht, indem Basic den Interruptvektor 8 auf eine Routine verzweigt, die nun nur noch jedes vierte Mal das BIOS anspricht.

Es gibt zwei grundsätzliche Verfahrensweisen der Soundprogrammierung

Um nun einen Ton zu erhalten, müssen Sie den Soundchip veranlassen, eine Frequenz zu erzeugen. Dabei ist zu beachten, daß der Soundchip zwar jede gewünschte Frequenz zustandebringen kann, daß er aber für die Tondauer über keine eigenen Informationen verfügt. Es ist also die Aufgabe des Programms, dafür zu sorgen, daß der Ton wieder abgeschaltet wird.

Damit der Soundchip eine Frequenz erzeugen kann, müssen Sie eigene »clock ticks« generieren, indem Sie den

Soundchip mit einer Zahl versorgen. Der Soundchip zählt die Zyklen des Prozessors 8284 mit. Sind vorgegebene Zahl und die hochgezählte Anzahl des 8284-Prozessors identisch, wird ein Impuls ausgegeben, und der Vorgang wiederholt sich von neuem.

Der Soundchip unterteilt also die Uhrfrequenz, die 1,193 MHz beträgt, in gleiche Stücke, um die gewünschte Frequenz zu erhalten. Mit anderen Worten, der Soundchip sendet eine Serie von Impulsen aus, die einen Ton mit einer bestimmten Frequenz erzeugen. Diese Tonfrequenz wiederum errechnet sich aus der Uhrfrequenz (also 1,193 MHz) geteilt durch die vorgegebene Zahl. Diese vorgegebene Zahl darf sich im Bereich von 1 bis 65535 bewegen. Teilen Sie die Uhrfrequenz durch die größtmögliche Zahl (also 65535), dann erhalten Sie als Tonfrequenz einen Wert von 18,2 Hz, was ungefähr dem tiefen D (18,35 Hz) entspricht. Diese errechnete Zahl für die Tonfrequenz senden Sie zu den Registern des Soundchips und zwar mit der Übertragung von drei Werten.

Als erstes übertragen Sie den Wert 182 zu Port 67. Dieser Wert signalisiert dem Soundchip, daß als nächstes die errechnete Zahl folgt. Da es sich hier um einen Wert von der Länge eines Wortes (entspricht 2 Byte) handelt, schickt man zuerst das niederwertige Byte gefolgt vom höherwertigen Byte zur Portadresse 66.

Nachdem so der Soundchip mit der gewünschten Frequenz versorgt ist, ist nun noch der Lautsprecher zu aktivieren. Den Lautsprecher kontrolliert die programmierbare Peripherieschnittstelle (PPI), die mit der Portadresse 97 angesprochen wird. Allerdings dürfen Sie hier nur die zwei niedrigsten Bits ändern, da die restlichen sechs Bit des Byte für andere Zwecke verwendet werden. Das niedrigste Bit, also Bit 0, kontrolliert das Signal, um den Lautsprecher anzutreiben. Das zweite Bit kontrolliert das Pulsieren der Lautsprechermembrane (was ja inzwischen der Soundchip erledigt). Damit der Lautsprecher überhaupt auf die Signale des Soundchips reagiert, müssen Sie diese beiden Bit setzen (also mit dem Wert 1 versehen).

Es gibt aber noch eine andere Alternative, nämlich den Lautsprecher auch direkt anzusprechen. Hierzu senden Sie die Signale direkt zum PPI und umgehen somit den Soundchip. Als erstes aktivieren Sie den Lautsprecher, indem Sie Bit 0 des Ports 97 (die Adresse für den PPI) auf 0 setzen. Als nächstes setzen Sie Bit 1 abwechselnd auf 0 und 1, um die Membrane des Lautsprechers zum Schwingen zu bringen.

Mit dieser Methode können Sie auf sehr vielfältige Weise Töne generieren, jedoch sind die Nachteile nicht zu übersehen. Für die Höhe der Frequenz ist mit dieser Methode nur noch der Computer verantwortlich. Ein AT würde beispielsweise mit ein und demselben Programm einen höheren Ton erzeugen als ein PC/XT. Außerdem müssen, um einen guten und glatten Ton zu erhalten, die »clock ticks« ausgeschaltet werden. Dies führt wiederum dazu, daß Sie die damit zusammenhängenden Interrupts verwalten müssen. Der einzige Vorteil dieser Methode liegt darin, polyphone also mehrstimmige Melodien erzeugen zu können. Ein grundsätzlicher Nachteil der Soundprogrammierung bei PCs ist die Tatsache, daß keine Kontrolle der Lautstärke möglich ist. Dies liegt an der Qualität des Lautsprechertyps, der im PC eingebaut ist, und die meilenweit von der Qualität eines Hifi-Lautsprechers entfernt ist. Deshalb kann die Lautstärke je nach Frequenz verschieden groß sein.

Es ist sicherlich richtig, daß das andauernde Piepsen bei Programmen nach längerer Zeit auf die Nerven geht. Andererseits erspart man sich bei eingabe-intensiven Anwendungen eine Menge unnötiger Tipparbeit, wenn man akustisch auf einen Fehler aufmerksam gemacht wird. Ein Lösungsweg für solche Programme wäre der Einbau verschiedener (und angenehmerer) Töne, je nach Fehlerursache.

(gh)

Der PC spricht

Die Ausgabe von menschlicher Sprache durch den Computer war schon immer ein faszinierendes Thema. Mit einer kleinen Schaltung und einem GW-Basic-Programm können auch Sie Ihren PC zum Sprechen bringen.

Die meisten bislang veröffentlichten Bauanleitungen zum Thema Sprachausgabe waren immer mit einem großen Nachteil verbunden: Entweder war die dazu erforderliche Hardware sehr teuer und kompliziert, oder der Speicherbedarf für die digitalisierte Sprache war enorm.

Aus diesem Grund hat DOS International das Sprach-Ein-/Ausgabe-System »PC-Voice« entwickelt, das mit einem Minimum an Hard- und Software auskommt, und trotzdem eine erstaunlich verständliche Sprachwiedergabe bietet. Dabei ist der Speicherbedarf mit weniger als zwei KByte pro Sekunde derart gering, daß auch eine längere Ausgabe problemlos im Speicher des PC Platz findet. Zunächst aber eine Erklärung, wie sich Sprache überhaupt in einem Computer speichern läßt: Wenn Sie in ein Mikrofon sprechen, wandelt dieses die Schwingungen der Luft in elektrische Schwingungen um. Diese elektrischen Schwingungen lassen sich verstärken und dann über einen Lautsprecher wieder ausgeben. Die elektrischen Schwingungen des Mikrofons können nun aber nicht direkt im PC gespeichert werden, denn wie jedes Tonsignal ist auch das vom Mikrofon abgegebene Signal ein Analogsignal. Mit analogen Größen kann Ihr PC aber überhaupt nichts anfangen, denn er arbeitet ausschließlich mit den digitalen Signalen 0 und 1, die durch die beiden Spannungen 0 Volt und + 5 Volt dargestellt werden.

Um eine analoge Spannung nun in einen digitalen Wert umzuwandeln, verwendet man einen A/D-Wandler. Solche A/D-Wandler gibt es in den verschiedensten Ausführungen, die sich in Geschwindigkeit und Auflösung der Wandlung unterscheiden. Von der Auflösung hängt es ab, wie genau das Ergebnis ist. Die Auflösung wird in Bit angegeben und sie bestimmt, wie viele verschiedenen Werte der A/D-Wandler innerhalb seines Eingangsspannungsbereichs unterscheidet.

So gibt zum Beispiel ein 8-Bit-A/D-Wandler einen 8-Bit Wert (also ein Byte) zurück und unterscheidet daher $2^8 = 256$ verschiedene Werte.

Die Umwandlung einer analogen Größe in einen digitalen (binären) Wert nennt man auch Digitalisierung. Das beste Verfahren, menschliche Sprache mit einem Computersystem zu speichern, besteht nun darin,

das analoge Tonsignal dem Eingang eines A/D-Wandlers zuzuführen, den Wandler laufend abzufragen und das Ergebnis im Arbeitsspeicher des Computers abzulegen. Um die so digitalisierte Sprache wieder auszugeben, verwendet man einen D/A-Wandler, der die gespeicherten Zahlenwerte wieder in die entsprechende Spannung umwandelt.

Für die Digitalisierung von Sprache existieren verschiedene Verfahren, die zum Teil sehr aufwendig sind

Sehr hochwertige Digitalisierungssysteme verwenden dieses Verfahren, das allerdings einen großen Nachteil hat: Der Bedarf an Speicherplatz ist enorm. Es hat sich nämlich herausgestellt, daß die obere Grenzfrequenz, also die höchste Frequenz, die auf diese Weise wiedergegeben werden kann, der Hälfte der Anzahl von Wandlungen pro Sekunde entspricht. Im Klartext bedeutet dies, daß Sie 8000 Werte pro Sekunde speichern müssen, um Frequenzen bis 4000 Hertz wiederzugeben.

Bei einem A/D-Wandler mit 8-Bit-Auflösung beträgt der Speicherbedarf immerhin 8000 Byte, also fast 8 KByte pro Sekunde. Außerdem sind A/D- und D/A-Wandler immer noch recht teuer. Insgesamt ist der Aufwand bei diesem Verfahren sehr hoch.

Bei der Entwicklung von PC-Voice war das Ziel, eine ausreichende Verständlichkeit zu erreichen, die einerseits ohne aufwendige Zusatzschaltungen auskommt und andererseits weniger Speicherplatz verbraucht. PC-Voice arbeitet prinzipiell in der gleichen Art und Weise wie das oben beschriebene Verfahren, allerdings in wesentlich vereinfachter Form. Statt eine vollständige A/D-Wandlung durchzuführen, prüft dieses Verfahren nur, ob die anliegende Spannung kleiner oder größer als eine feste Vergleichsspannung ist. Auf diese Weise läßt sich zwar nur feststellen, ob eine positive oder eine negative Halbwelle am Eingang anliegt, die Wellenform geht dabei also verloren. Bei der Ausgabe entstehen jedoch Rechteckschwingungen, deren Frequenz weitgehend der des Eingangssignals entspricht. Für eine verständliche Wiedergabe reicht diese »1-Bit-A/D-Wandlung« daher schon aus.

Dieses Verfahren ist allerdings nicht ganz neu, denn es existieren Programme für viele Heimcomputer, die Sprache und Geräusche auf die gleiche Art und Weise über den Kassettenrecorder-Eingang digitalisieren und über den Lautsprecher des Heimcomputers wiedergeben. Diese Heimcomputer-Programme benötigen daher keine zusätzliche Hardware.

Das Problem bei einer Umsetzung für PCs liegt darin, daß heute kaum ein PC mehr ein Kassettenrecorder-Interface besitzt. Um das Prinzip der 1-Bit-A/D-Wandlung zur Sprachdigitalisierung auf dem PC trotzdem anwenden zu können, müssen Sie die Funktion eines Kassettenports nachbilden. Im einfachsten Fall besteht die einzige Aufgabe eines Kassettenports darin festzustellen, ob an seinem Eingang eine positive oder eine negative Halbwelle anliegt.

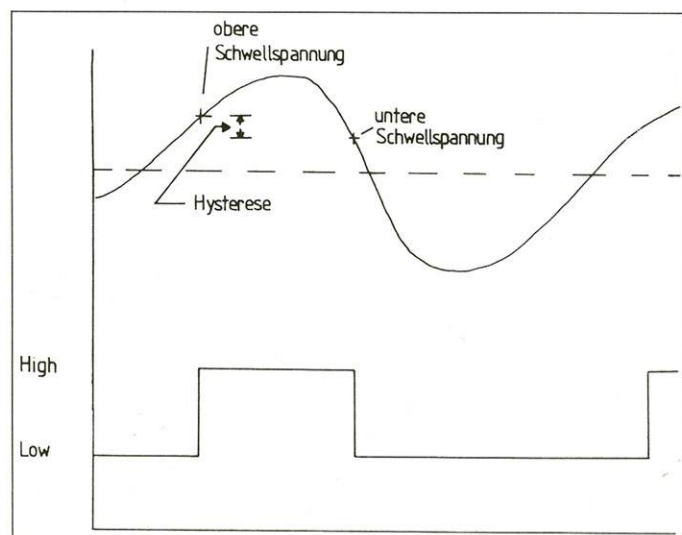


Bild 1. So funktioniert die 1-Bit-A/D-Wandlung

Zu diesem Zweck enthält eine solche Schaltung einen Logikbaustein mit »Schmitt-Trigger«-Eingang, das heißt, im Gegensatz zu einem normalen Logik-Eingang interpretiert dieser Baustein ein Signal dann als 1, also gesetzt, wenn es eine bestimmte Schwellspannung überschritten hat. Wenn die Spannung danach wieder absinkt, liegt die Schwelle zum 0-Pegel unterhalb des Pegels, bei dem der Umschaltpunkt zum 1-Pegel war.

Mit anderen Worten, ein Schmitt-Trigger-Eingang verhält sich wie ein »Knackfrosch«: Wenn man einen solchen Knackfrosch langsam durchdrückt, knackt er an einem bestimmten Punkt um. Wenn Sie ihn darauf langsam wieder loslassen, springt er nicht wieder am ersten Knackpunkt in den Urzustand zurück, sondern erst etwas später.

Für die Konstruktion eines 1-Bit-A/D-Wandlers genügt ein einziger Schmitt-Trigger

Auf diese Weise verhindert der Schmitt-Trigger ein undefiniertes Hin- und Herpendeln zwischen dem 0- und dem 1-Signal, wenn das Eingangssignal gerade den zum Umschalten nötigen Pegel aufweist. Die Differenz zwischen oberer und unterer Schwellspannung nennt man Hysterese. Bild 1 zeigt, wie ein Schmitt-Trigger ein analoges Eingangssignal (oben) in ein digitales Ausgangssignal (unten) umwandelt.

Dieses Wissen läßt sich nutzen, um an der Druckerschnittstelle des PC einen Kassettenport nachzubilden. Dazu bedienen Sie sich des integrierten Bausteins 4093. Dieses CMOS-IC im 14poligen Gehäuse enthält vier Nand-Gatter mit Schmitt-Trigger-Eingängen. Bild 2 zeigt die Anschlußbelegung des 4093 von oben.

In Bild 3 sehen Sie den Schaltplan für die Digitalisierungsplatine und die Pinbelegung der beiden verwendeten Transistoren von unten. Am Eingang dieser Schaltung befindet sich eine einfache Verstärkerstufe auf Basis des Transistors BC548. Sie erfüllt zwei Aufgaben: Sie verhindert einmal, daß negative Spannungen oder Spannungen, die größer als die Versorgungsspannung des CMOS-ICs sind, den Eingang des 4093 erreichen.

Ein CMOS-Baustein ist zwar sehr flexibel in der Höhe seiner Betriebsspannung, doch führen negative Spannungen oder Spannungen, die seine Versorgungsspannung überschreiten, sehr leicht zur Zerstörung des Bausteins. Andererseits bringt

die Transistorstufe das eingehende Signal auf den Pegel, der für die Digitalisierung ideal ist.

Das beste Ergebnis erzielt man, wenn der Nulldurchgang bei der Hälfte der Versorgungsspannung liegt. Die beiden Kondensatoren befreien das Niederfrequenzsignal von überlagernder Gleichspannung, die die Digitalisierung verfälschen. Unter Verwendung des Trimmers läßt sich der dem Schmitt-Trigger zugeführte Pegel in weiten Grenzen verändern.

Über ein Trimpotentiometer läßt sich die Digitalisierungsschaltung optimal einstellen

Die eigentliche Digitalisierung erledigt das erste Nand-Gatter, dessen beiden Eingänge über zwei Widerstände auf der Hälfte der Betriebsspannung liegen. Das zweite Nand-Gatter dient als Inverter, es macht also die Invertierung, die das erste Gatter verursacht, wieder rückgängig. Über einen 1-Kiloohm-Widerstand ist der Ausgang dieses Gatters mit Pin 11 (Busy-Leitung) der Druckerschnittstelle verbunden.

Normalerweise benutzt der Drucker diese Leitung, um dem PC zu signalisieren, daß er mit dem Ausdrucken von Daten beschäftigt ist und daher keine weiteren Daten annehmen kann. In der Digitalisierungsschaltung wird die Busy-Leitung dagegen zur Abfrage des Pegels mißbraucht. Bei dieser Gelegenheit stellt sich natürlich die Frage, unter welcher Adresse diese Leitung ansprechbar ist. Die Busy-Leitung entspricht Bit 7 des Statusregisters der jeweiligen Druckerschnittstelle, wobei die Adresse des Statusregisters immer die auf die Basisadresse der Druckerschnittstelle folgende Adresse ist. Beim PC sind drei verschiedene Basisadressen für Druckerschnittstellen zulässig: 03BChex, 0378hex und 0278hex. Oft werden diese Basisadressen mit den MS-DOS-Bezeichnungen »lpt1«, »lpt2« beziehungsweise »lpt3« gleichgesetzt. In Wirklichkeit dürfen Sie aber nicht davon ausgehen, daß die Druckerschnittstelle, die Sie unter MS-DOS als »lpt1« ansprechen, der Basisadresse 03BChex entspricht.

Vielmehr verfährt ein PC nach dem Einschalten wie folgt: Zunächst wird geprüft, ob ein Monochromadapter oder eine Hercules-Grafikkarte mit Druckerschnittstelle im Computer installiert ist. Die Basisadresse einer solchen Druckerschnittstelle ist immer 03BChex. Wenn der PC eine solche Schnittstelle findet, deklariert er diese als »lpt1«, wenn nicht, weist er »lpt1« die Basisadresse 0378hex zu.

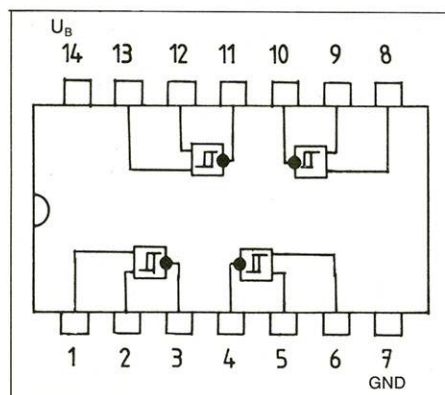


Bild 2. Der Baustein 4093 enthält vier Nand-Gatter mit Schmitt-Trigger-Eingängen

Falls eine weitere Druckerschnittstelle (Basisadresse 0278hex) im Computer vorhanden ist, entspricht diese »lpt3«, sofern ein Monochromadapter oder eine Hercules-Grafikkarte installiert ist, ansonsten wird ihr die Bezeichnung »lpt2« zugewiesen. Die Basisadresse der Druckerschnittstelle auf Multifunktionskarten läßt sich meistens wahlweise auf 0378hex oder 0278hex einstellen.

Ein Lautsprecher in der Digitalisierungsschaltung erlaubt das Mithören während des Digitalisierens

Das dritte Nand-Gatter des 4093-Bausteins ist parallel zum zweiten Nand-Gatter geschaltet und macht ebenso wie das zweite die Invertierung des ersten Gatters wieder rückgängig. Über einen Vorwiderstand wird dieses Signal dem Transistor BC 108 zugeführt, der das digitalisierte Signal zur Kontrolle auf den angeschlossenen Lautsprecher ausgibt. Auf diese Weise können Sie das Ergebnis der Digitalisierung in Echtzeit hören und gegebenenfalls die Stellung des Trimmers optimieren, ohne dazu den PC zu bemühen.

Ein wichtiger Aspekt beim Nachbau der Digitalisierungsschaltung ist die Spannungsversorgung, denn sie ist auch für die Qualität der Digitalisierung entscheidend. Der Baustein 4093 ist im Prinzip ziemlich anspruchslos in der Höhe der Versorgungsspannung, er arbeitet mit Spannungen zwischen 3 und 12 Volt zufriedenstellend. Weil aber seine Ausgangsspannung, also die Spannung, die am Ausgang eines Gatters bei dem Pegel 1 anliegt, in etwa der dem Baustein zugeführten Versorgungsspannung entspricht, die Busy-Leitung aber ein TTL-Eingang ist, der nur mit 0 und + 5 Volt arbeitet, liegt

bild dieses Verstärkers sehen Sie in Bild 6.

Der integrierte Baustein LM 386 von National Semiconductor bildet das Herz dieser Schaltung. Tabelle 2 listet die Bauteile auf, die für den Verstärker erforderlich sind, und Bild 7 zeigt ein Muster des Niederfrequenz-Verstärkers. Der Aufbau dieser Schaltung ist so unkompliziert, daß sich weitere Erklärungen erübrigen. Vergessen Sie aber nicht, daß auch der LM 386 gegen zu starke Erwärmung während des Lötens empfindlich ist. Verwenden Sie also auch für dieses IC eine Fassung.

Die Versorgungsspannung des Verstärkers muß zwischen 5 und 12 Volt liegen, prinzipiell können Sie den Verstärker also aus der gleichen Stromquelle speisen wie die Digitalisierungsplatine. Für den Verstärker lohnt sich aber eventuell ein Stecker-Netzteil, denn bei voller Aussteuerung verbraucht der LM386 schon 100 Milliampere Strom oder mehr. Eine 9V-Blockbatterie würde diese Belastung nur etwa eine halbe Stunde durchhalten.

Mit dem Niederfrequenz-Verstärker erhalten die digitalisierten Signale die richtige Lautstärke

Den Niederfrequenz-Verstärker verbinden Sie am besten über einen Cynchstecker mit dem Ausgang einer beliebigen Signalquelle, wie beispielsweise Kassetten-Recorder, HiFi-Anlage oder Mikrofon mit nachgeschaltetem Verstärker. Schalten Sie die Versorgungsspannung ein und testen Sie den Verstärker, indem Sie an seinen Ausgang einen Lautsprecher anschließen.

Wenn Sie nun den Verstärker mit der Signalquelle und der Digitalisierungsplatine verbinden und danach die Versorgungsspannung(en) einschalten, hören Sie im Lautsprecher der Digitalisierungsplatine die digitalisierte Form des eingespeisten Tonsignals. Eventuell müssen Sie das Trimpotentiometer des Verstärkers verstellen, um das Ergebnis zu verbessern.

Sie erhalten nun einen ersten Eindruck von der Qualität der Digitalisierung. Wer auf CD-Qualität gehofft hat (nach dem Motto: Die CD-Aufzeichnung ist doch auch digital), wird leider enttäuscht. Das Ergebnis ist eher mit einem übersteuerten Mittelwellenradio zu vergleichen. Gute Sprachaufnahmen sind aber klar verständlich. Um die Qualität zu steigern, verwenden Sie bitte Höhen- und Tiefenregler, um die besonders hohen und besonders tiefen Frequenzen zu dämpfen. Sehr gut eignet sich

dazu ein Equalizer, den Sie so einstellen, daß er sich wie ein Bandpaß verhält, das heißt, sie lassen nur den reinen Sprachbereich (etwa zwischen 100 und 3000 Hz) zum Verstärker und blenden die übrigen Frequenzen aus. Für das Abspeichern der digitalisierten Tonsignale im PC ist die Software von PC-Voice zuständig. Die Steuersoftware muß die Busy-Leitung der Druckerschnittstelle laufend abfragen und das Ergebnis im Arbeitsspeicher des PC ablegen. Bei Bedarf liest das Programm die gespeicherten Daten im Speicher und gibt sie über den Lautsprecher des PC wieder aus. Das hat den Vorteil, daß für die Wiedergabe der Sprache keine zusätzliche Hardware nötig ist, das heißt, ein auf diese Weise mit Sprache ausgestattetes Programm läuft auf jedem beliebigen PC.

Die digitalisierte Sprache läßt sich problemlos in jedes andere Basic-Programm einbinden

Wie bereits eingangs beschrieben wurde, müssen bei der Digitalisierung eines Tonsignals mindestens 8000, besser 10000 und noch besser 12000 Abfragen in der Sekunde gespeichert werden. Da jede Hochsprache für diese Aufgabe zu langsam ist, muß die eigentliche Abfrageroutine in Assemblersprache geschrieben sein. Um den Programmieraufwand trotzdem gering zu halten, ist das Steuerprogramm von PC-Voice in GW-Basic geschrieben und verwendet nur für die eigentliche Digitalisie-

rung und Wiedergabe der Sprache eine Unteroutine in Assembler.

Prinzipiell ist das Einbinden von Assembler-Unterprogrammen in Basic nicht weiter schwierig, wenn dieser Vorgang auch unzureichend dokumentiert ist. Das Problem dabei ist nur, daß eine Assembler-Unteroutine nur relative Adressen enthalten darf, das heißt, es sind keine absoluten Sprünge oder Speicherzugriffe auf absolute Adressen zulässig. Die Assembleroutine für PC-Voice muß aber auf einen reservierten Speicherbereich zugreifen, um dort die digitalisierten Daten abzulegen beziehungsweise auszulesen.

Die Lösung zu diesem Problem ist eigentlich nicht schwer, jedoch nirgends dokumentiert, so daß PC-Voice unseres Wissens nach das erste GW-Basic-Programm beinhaltet, das eine Assembleroutine mit absolutem Speicherzugriff enthält. Dazu muß man einfach alle absoluten Speicherzugriffe auf die spätere Arbeitsumgebung anpassen, das heißt in diesem Fall zu der vom Assembler erzeugten Adresse des Speichersbereichs für die digitalisierten Daten noch den Offset der Startadresse addieren.

Damit Sie digitalisierte Sprache auch in Ihren eigenen Basic-Programmen verwenden können, werden 32 KByte des Speichers, der Basic zur Verfügung steht, mit dem Befehl »clear ,32767« für die Zwecke von PC-Voice reserviert. Nach dem Programmstart überträgt das Basic-Programm dann unter Verwendung des Poke-Befehls das Assemblerprogramm aus den Data-Zeilen in den Speicher. Da die Unteroutine nur 172 Byte lang ist,

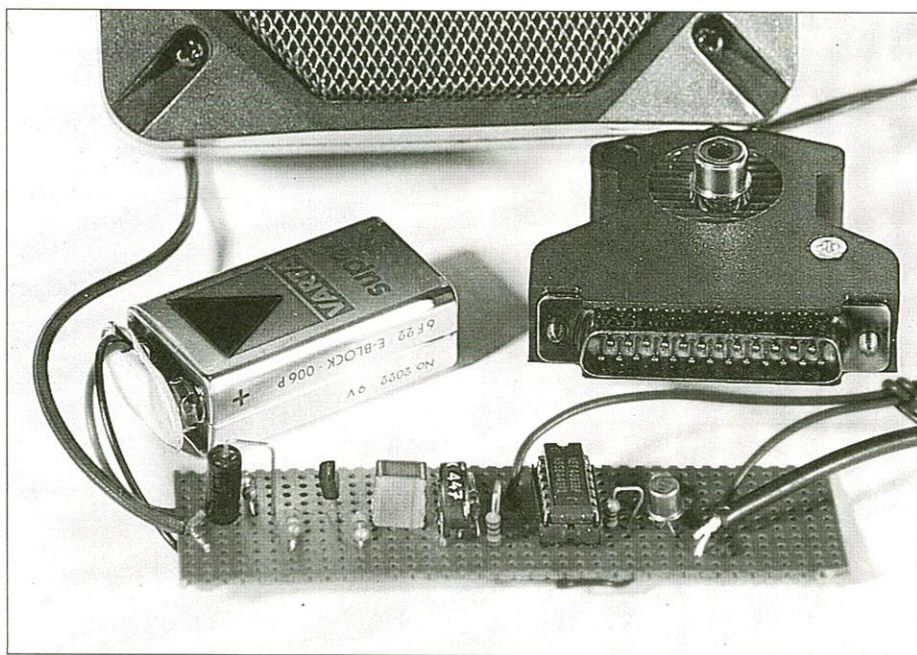


Bild 4. Ein Muster der Digitalisierungsschaltung mit den erforderlichen Anschlüssen, einer Batterie und Lautsprecher

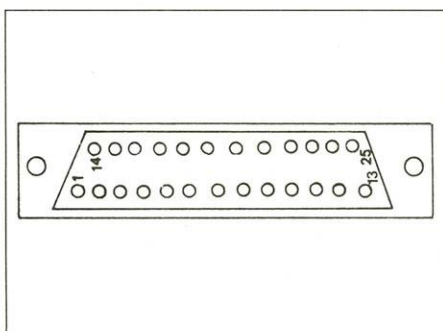


Bild 5. Die Pinbelegung des 25-poligen DSUB-Steckers (Ansicht von der Lötseite)

stehen so noch über 32000 Bytes für die Daten zur Verfügung, in dem selbst bei relativ hoher Abtastrate fast eine halbe Minute synthetischer Sprache Platz findet.

Mit Hilfe der Befehle »bsave« und »bload« läßt sich der 32 KByte große Speicherblock, also Programm und digitalisierte Daten, auf Diskette schreiben und in jedes andere Basic-Programm, das zuvor den erforderlichen Speicher reserviert hat, einbinden.

Mit einer kleinen Schaltung läßt sich die Sprache vom PC-Lautsprecher auf die HiFi-Anlage umleiten

Bevor Sie jedoch genauer die Bedienung und den Ausbau des Steuerprogramms sowie die Einbindung in eigene Programme erfahren, erst noch eine kleine Bauanleitung, mit der Sie die Sprachausgabe vom PC-Lautsprecher auf Ihre HiFi-Anlage oder einen anderen Niederfrequenz-Verstärker umleiten können. Die Schaltung zur Ausgabeumleitung ist besonders für diejenigen Leser nützlich, deren PC

nur einen mittelmäßigen Lautsprecher besitzt. HiFi-Sound dürfen Sie dennoch nicht erwarten, wenngleich die Ausgabequalität beträchtlich erhöht wird.

Bild 8 zeigt die Schaltung zur Wiedergabe der Sprache über die HiFi-Anlage. Sie wird ebenso wie die Digitalisierungsplatine an der Druckerschnittstelle angeschlossen, und zwar am ersten Datenbit (Bit 0). Dieses Bit läßt sich ein- und ausschalten, indem Sie Bit 0 der Basisadresse der betreffenden Druckerschnittstelle setzen beziehungsweise löschen.

Das Prinzip dieser Schaltung ist sehr einfach zu verstehen: Der Spannungsteiler, den die beiden Widerstände bilden, setzt die Ausgangsspannung von dem TTL-Pegel (+ 5 Volt) auf etwa 1,8 Volt herab. Der 100-nF-Kondensator dient zur gleichspannungsmäßigen Entkopplung.

Schließen Sie an den Ausgang dieser Schaltung jedoch nur Verstärker mit einer Cynch-Eingangsbuchse an, denn nur für diese ist der 1,8 Volt-Pegel geeignet. Versuchen Sie nicht, anstelle der Cynchbuchse eine DIN-Steckverbindung anzuschließen, denn für einen DIN-Eingang ist der Spannungspegel von 1,8 Volt zu hoch.

Möchten Sie die Schaltung dennoch an ein Gerät mit DIN-Eingang anschließen, so müssen Sie den 10-Kiloohm-Widerstand durch einen 10-Kiloohm-Trimmregler und den 33 kOhm-Widerstand durch einen 100-Kiloohm-Widerstand ersetzen. Mit dem Trimmregler passen Sie den Ausgangspegel dann dem optimalen Eingangspegel an.

Diese simple Schaltung bauen Sie am besten in das Steckergehäuse des 25-poligen DSUB-Steckers ein und lassen die Cynchbuchse seitlich her-

ausschauen, wie es auch in Bild 4 zu sehen ist.

Weil sich die Wiedergabe über die HiFi-Anlage von der Ausgabe am PC-Lautsprecher stark unterscheidet, enthält das PC-Voice-Programm (Listing 1) eine eigene Routine für die Ausgabe über einen externen Verstärker. Der erste Teil dieses Programms beschränkt mit dem Befehl »clear, 32767« (das Komma ist wichtig) den Speicherbereich, der GW-Basic zur Verfügung steht, auf 32767 Byte, reserviert also die restlichen 32 KByte für die Unteroutine und die digitalisierten Daten.

Danach überträgt das Programm die Assembleroutinen aus den Data-Zeilen in den reservierten Speicherbereich und überprüft dabei gleichzeitig die Data-Zeilen auf Eingabefehler. Anschließend werden den Variablen die Einsprungsadressen für die drei Teile der Assembleroutine zugewiesen und zuletzt wird die Fehlerbehandlungsroutine aktiviert. Damit ist der Initialisierungsteil des Programms abgeschlossen.

Das Steuerprogramm für PC-Voice ist in GW-Basic und Assemblersprache geschrieben

Es folgen der Aufbau des Titelbildes und des Hauptmenüs. Für jeden Menüpunkt sind eigene Unterprogramme vorhanden, die aufgrund der ausführlichen Dokumentation keiner weiteren Erläuterung bedürfen. Das ganze Basic-Programm macht intensiven Gebrauch von der Unterprogramntechnik und ist durch diese Aufteilung in überschaubare Teile leichter zu verstehen. Erwähnenswert sind noch die Hilfsroutinen ab Zeile 5000, in denen häufig verwendete Funktionen, wie die zentrierte Ausgabe einer Zeichenkette oder das Warten auf einen Tastendruck, enthalten sind und die im Hauptprogramm durch einen einfachen Gosub-Befehl aufgerufen werden.

Der letzte Teile des Basic-Programms enthält eine Routine zur Fehlerbehandlung. Dadurch führen während des Programmablaufs auftretende Fehler, wie zum Beispiel eine offene Laufwerkklappe, nicht zu einer Unterbrechung des Programms.

Listing 2 zeigt den Quellcode der Assembleroutinen. Der Hauptteil des Programms besteht aus nur drei Sprungbefehlen, die am Anfang des Programms stehen. Jeder Sprungbefehl ruft einen der drei Programmteile (Ausgabe über den PC-Lautsprecher, Ausgabe über die HiFi-Anlage und Digitalisierung) auf. Dadurch wird erreicht, daß das Basic-Steuerprogramm jede einzelne Routine sehr

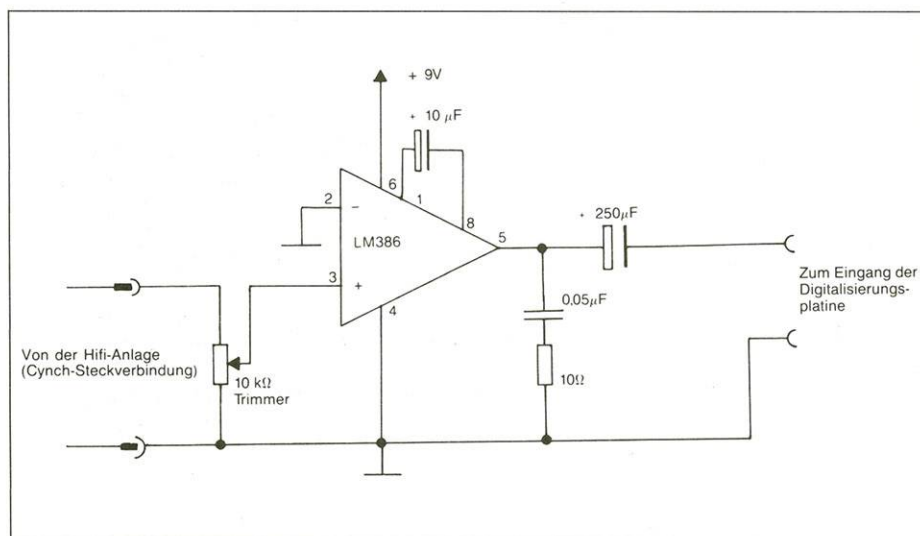


Bild 6. Dieser einfache Niederfrequenz-Verstärker ersetzt Ihre HiFi-Anlage

leicht aufrufen kann, denn weil ein Sprungbefehl immer drei Byte lang ist, führt ein Aufruf der Startadresse (32768) zum Start der Wiedergaberoutine über den Lautsprecher, ein Aufruf von Startadresse + 3 zum Start der Wiedergaberoutine über die HiFi-Anlage und ein Aufruf von Startadresse + 6 zum Start der Digitalisierungsroutine.

Die drei Routinen selbst sind als Far-Prozeduren deklariert, damit der Assembler den abschließende Return-Befehl (Assemblerbefehl »ret«) als Far-Return codiert (Assemblerbefehl »retf«). Bei einem Far-Return holt der Prozessor beim Rücksprung sowohl das Segment als auch den Offset vom Prozessorstapel. Ein solches Far-Return ist für den korrekten Rücksprung zu GW-Basic erforderlich.

Die Assembler Routinen selbst erfüllen eher einfache Aufgaben: Die Digitalisierungsroutine fragt laufend den Zustand der Busy-Leitung ab, setzt dementsprechend das Bit 0 des AH-Registers, rotiert dieses Register insgesamt achtmal, legt es dann im für die Daten reservierten Speicherplatz ab und wiederholt dies, bis das Ende des Datenspeichers erreicht ist.

Die im GW-Basic-Programm integrierten Assembler Routinen sorgen für Geschwindigkeit

Die beiden Wiedergaberoutinen lesen ein Byte aus dem Datenspeicher aus, fragen die Bits nacheinander ab und schalten den Lautsprecher beziehungsweise das für die HiFi-Anlage zuständige Bit ein oder aus.

Alle drei Routinen rufen bei jedem Schleifendurchlauf eine Warteroutine auf, um die Abtastrate auf das erforderliche Maß zu bremsen.

Das Programm bietet weiterhin die Option an, die Wiedergabegeschwindigkeit und die Länge der Aufzeichnung sowie die Basisadresse der Druckerschnittstelle zu verändern. Das ist zum Beispiel dann nützlich, wenn Sie Sprache auf einem PC mit 4,77 MHz Taktfrequenz digitalisieren und auf einem AT bei 8 MHz wiedergeben möchten. Würde in diesem Fall die Ablaufgeschwindigkeit nicht verändert, so wäre die Wiedergabe nicht verständlich, da sie etwa sechsmal schneller erfolgt als auf dem PC.

Die Änderung des Geschwindigkeitsparameters erledigt das Programm auf eine ungewöhnliche aber auch sehr effiziente Weise. Das Verfahren läßt sich am besten als »dynamisches Patchen« bezeichnen, denn das Basic-Programm manipuliert unter Verwendung der Basic-Befehle »peek« und »poke« direkt den Maschinencode der Assembler Routinen.

Das PC-Voice-Programm arbeitet völlig menügesteuert und ist selbst von Computer-Einsteigern problemlos zu bedienen. Bitte bedenken Sie jedoch, daß sich der Ablauf der Assembler-Unterprogramme nicht mit [Ctrl-Scroll Lock] unterbrechen läßt, da es sich um reinen Maschinencode handelt.

Mit den Menüpunkten 4 und 5 können Sie die derzeit im Arbeitsspeicher befindliche Aufzeichnung auf die Diskette beziehungsweise Festplatte speichern und wieder einlesen. Zuvor zeigt Ihnen das Programm auf



... make it easy!

Neueste Software-Version von der COMDEX, USA

PC TOOLS de Luxe

(Neu: Caching, Text, Data-Recovery, Tuning u.v.m.) mit dt. Handbuch DM 188,10

COPY II PC

(auch für 3 1/2 ") Kopierprogramm DM 111,70

OPTION BOARD

Neu! Deluxe (auch für 3 1/2 ") Mac Intosh DM 399,-

1 DIR Plus

Benutzeroberfläche DM 313,50

Automenu

Menügenerator DM 199,50

BackEZ

Datensicherung MS-DOS-Format (auch für LAN u. Floppy-Tape) dt. Handbuch DM 298,00

PrintQ

Druck-Spooling auf Platte DM 273,60

PrintMate

res, Druckersteuerung (vertikal) DM 210,90

LaserMate

res. Druckersteuerung f. Laserdrucker DM 313,50

Brooklyn Bridge

Desk-Top-Lap-Top, 5 1/4 " - 3 1/2 ", ser. Kommunikation, einschl. Kabel DM 399,00

Easy Flow

Flowchart-Generator DM 427,50

REFEREE

Verwaltung res., Software DM 256,90

Personal VM

Benutzeroberfläche für VM/CMS DM 883,50

DESQview 2.0

Multitasking u.v.m. incl. QEMM 386 DM 399,00

The DOCUMENTATOR

(Wallsoft) Analyse u. Dokumentation f. dBase, Clipper, Fox-Base, dt. Handbuch DM 855,00

SPF/PC 2.0

Editor MVS/TSO (Neu: Split-Screen, 43/50 Zeilenlänge, Macros wie TSO 2.2) orig. engl. DM 701,10

(in Vorbereitung) zus. dt. Handbuch DM 758,10

KEDIT 3.53

Editor VM/CMS DM 541,50

Pers. REXX 1.6

Interpreter u. Sprache (IBM-SAA) DM 541,50

GML/PC

(DCF/Script-VS f. Postscript-Laser) DM a. Anfr.

Fragen Sie ihren Händler oder direkt:

MCS Werner Kortmann

Wilh.-Reuter-Str. 12
6239 Eppstein
Tel. (0 61 98) 324 98

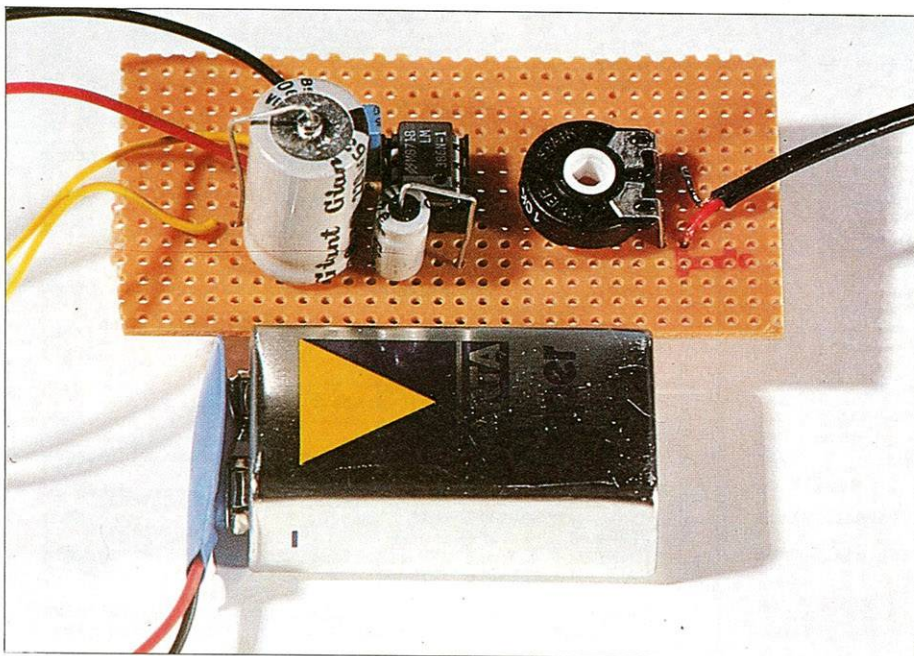


Bild 7. Auch der Niederfrequenz-Verstärker läßt sich platzsparend aufbauen

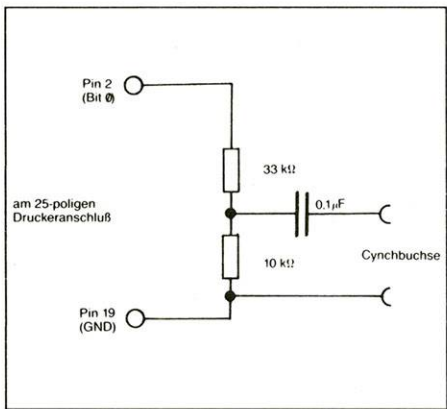


Bild 8. Wenn der PC-Lautsprecher Ihren Ansprüchen nicht genügt, läßt sich die Sprache mit dieser Zusatzschaltung auch über die HiFi-Anlage ausgeben

Wunsch das aktuelle Inhaltsverzeichnis an. Beim Speichern von digitalisierten Daten werden die aktuellen Parameter, also Ablaufgeschwindigkeit, Länge und Adresse der Druckerkarte mit abgelegt.

Der Menüpunkt 4 erlaubt Ihnen nicht nur, digitalisierte Sprache auf der Diskette oder Festplatte zu sichern, und später über PC-Voice wieder auszugeben. Mit diesem Menüpunkt erzeugen Sie auch 32 KByte lange Assembler-Unterprogramme, die sich in Ihre eigenen Basic-Programme einbinden lassen.

Dazu muß Ihr Basic-Programm inklusive Variablen kürzer als 32 KByte sein und folgende Anweisungen enthalten:

```
1000 clear ,32767
1010 bload "<dateiname> ",32768
1020 start=32768
```

Selbstverständlich lassen sich auch andere Zeilennummern verwenden. Die Sprachausgabe rufen Sie dann ganz einfach mit call start auf.

Auf diese Weise lassen sich auch beliebig viele digitalisierte Texte auf der Diskette speichern und zu einer akustischen Soundshow zusammenfügen.

Ein spezieller Menüpunkt erlaubt dem Benutzer das Ändern der Parameter für die Digitalisierung

Der weitaus wichtigste Menüpunkt des PC-Voice-Programms ist aber Punkt 6, »Parameter ändern«. Er führt Sie in ein Untermenü, in dem Sie wählen können, welche Parameter Sie verändern oder ob Sie ins Hauptmenü zurückkehren möchten. Das Parameter »Länge« gibt an, wieviele Bytes das Programm digitalisiert. Dadurch läßt sich zwar kein weiterer Speicherplatz für Basic-Programme (für Ihre eigenen Programme) schaffen, weil PC-Voice immer an den Offset von 32768 gebunden ist, zumindest können Sie damit aber die Zeitdauer der Wiedergabe abkürzen, denn während der Wiedergabe kann Ihr PC mit dem eigentlichen Basic-Programm nicht fortfahren. Der Maximalwert für Länge beträgt 32500 Bytes.

Die Zeitdauer der Aufzeichnung beziehungsweise Wiedergabe hängt aber nicht nur von der Länge, sondern auch vom zweiten Parameter, der Geschwindigkeit, ab. Hier sind Werte

zwischen 1 und 255 erlaubt, aber nicht alle Werte sind sinnvoll.

Von dem Geschwindigkeitswert hängt nämlich nicht nur die Geschwindigkeit der Wiedergabe (relativ zur Geschwindigkeit der Aufnahme) sondern auch die Anzahl der Abfragen pro Sekunde, also die Abtastrate ab. Wie Sie bereits erfahren haben, müssen Sie mindestens 8000 bis 12000 Abfragen pro Minute durchführen. Für die Warteschleife sind daher bei einem PC mit 4,77 MHz Taktfrequenz Werte von 1 bis 20 sinnvoll.

Bei einem 8-MHz-AT entspricht diese Abfragerate Werten zwischen 6 und 120, das heißt, eine Digitalisierung, die auf einem einfachen PC bei einem Geschwindigkeitswert von 7 vorgenommen wurde, muß auf einem 6-MHz-AT mit einer Geschwindigkeit von 42 wiedergegeben werden.

Weil die Geschwindigkeit der Aufzeichnung beim Sichern auf Diskette mitgespeichert wird, ergibt sich folgendes Problem: Wenn Sie Ihr eigenes Programm oder Ihre akustische Soundshow auf einem schnelleren oder langsameren Computer wiedergeben möchten, muß das Programm die Geschwindigkeit der Wiedergabe verändern. Dazu müssen Sie das »dynamische Patchen« in Ihr eigenes Programm integrieren, indem Sie den Befehl

poke 32933,<geschwindigkeit> einfügen. Natürlich läßt sich über diese Speicherstelle auch die im Programm gespeicherte Geschwindigkeit auslesen:

```
geschw=peek(32933)
```

(Martin Hepp/ma)

```
[R25] title pcvoice.asm
[V10] ;
[SAB] ;Assembler-Unterroutine für PC-Voice
[AB3] ;Version 2.0
[A16] ;(c) 1988 by Martin Hepp
[AV1] ; 30.03.1988
[V10] ;
[000] ;
[YX4] ;-----
[MV7] ; Konstantendefinition
[YX4] ;-----
[000] ;
[000] ;
[255] lpt1 equ 03BCh
[TT3] lpt2 equ 0378h
[VT3] lpt3 equ 0278h
[SL4] prnadr equ lpt2
[776] laenge equ 7ef4h
[378] pufadr equ offset memory+32768
[000] ;
[000] ;
[YX4] ;-----
[KU1] ; Code
[YX4] ;-----
[000] ;
[DPG] code segment para public 'code';Standard-Einleitung
[1NF] assume cs:code,ds:code ;nur ein Segment verwenden
[000] ;
[YX4] ;-----
[5M5] ; Sprungtabelle
```

```
[YX4] ;-----
[000] ;
[1WE] start: jmp output1 ;wenn Einsprung an der Start-
[N18] ;adresse, dann Ausgabe über
[NA8] ;den Lautsprecher aufrufen
[9PF] jmp output2 ;wenn Einsprung bei [star
[tadresse+3],
[P89] ;dann Ausgabe über Hifi-Anlage
[413] ;aufrufen
[DYF] jmp digitize ;wenn Einsprung bei [star
[tadresse+6],
[0C6] ;dann digitalisieren
[000] ;
[000] ;
[ZR3] ;-----
[MR7] ; Ausgabe am Lautsprecher
[ZR3] ;-----
[000] ;
[95G] output1 proc far ;Unterprogramm zur Ausgabe der
[11D] ;digitalisierten Daten am
PC-Lautsprecher
[Y39] push bp ;Base-Pointer sichern
[DGH] mov si,pufadr ;Anfangsadresse der digitalisierten Daten
[EP4] ;in Quellindex
[RCK] mov dx,laenge ;Länge der Digitalisierung (in Bytes) in das
```



```

[U24]
[ENA] loop5: push dx ;DX-Register und
[XCB] mov ch,08h ;DX-Register sichern
[EN8] mov ah,[si] ;insgesamt 8mal rotieren
[26E] loop6: in al,061h ;Datenbyte in AH
;Status von Port 61h einl
esen
[39A] and al,0fdh ;Bit 0 und Bit 1 löschen
[81B] test ah,80h ;oberstes Bit (Bit 7) tes
ten
[SMH] jz low2 ;wenn nicht, dann nächste
n Befehl überspringen
[Z06] or al,02h ;Bit 1 setzen
[86E] low2: out 061h,al ;AL wieder an Port 61h au
sgeben
[CC8] mov cl,01 ;einmal rotieren
[L58] cll ;Carry-Flag löschen
[D7D] rcl ah,cl ;AH-Register CL-mal links
rotieren
[33B] call warte ;Warterroutine aufrufen
[YDG] dec ch ;Rotationszähler um Eins
dekrementieren
[2LG] jnz loop6 ;wenn Zähler ungleich Nul
l, dann wiederholen
[SUF] inc si ;Zeiger auf Datenspeicher
um Eins erhöhen
[N6A] pop dx ;Länge vom Stapel holen
[51B] dec dx ;und um Eins dekrementier
en
[1YF] jnz loop5 ;wenn Länge ungleich Null
, dann weiterer
[7RC] ;Durchgang; ansonsten ist
die Wiedergabe
[3S2] ;beendet
[UGK] and al,0fdh ;sicherheitshalber Lautsp
recher ausschalten
[UK3] out 061h,al
[H7D] pop bp ;Register wieder vom Stap
el holen
[LD8] ret ;Rücksprung zu Basic
[UR3] output1 endp
[000]
[MR2] ;-----
[XX4] ; Ausgabe an der
[4N3] ; Hifi-Anlage
[MR2] ;-----
[000]
[G5G] output2 proc far ;Unterprogramm zur Ausgab
e der
[LHF] push bp ;digitalisierten Daten an
der Hifi-Anlage
[DGH] mov si,pufadr ;Anfangsadresse der dig
italisierten Daten
[EP4] ;in Quellindex
[RCK] mov dx,laenge ;Länge der Digitalisier
ung (in Bytes) in das
[UC3] ;DX-Register
[SNA] loop7: push dx ;DX-Register sichern
[XCB] mov ch,08h ;insgesamt 8mal rotieren
[EN8] mov ah,[si] ;Datenbyte in AH
[SGK] loop8: mov dx,prnadr ;Basisadresse der Druck
erkarte in DX
[WM0] nop
[81B] test ah,80h ;oberstes Bit (Bit 7) tes
ten
[WMH] jz low3 ;wenn nicht, dann nächste
n Befehl überspringen
[M86] mov al,01h ;Bit 1 setzen
[4XC] low3: out dx,al ;AL am Druckerport ausgeb
en
[ZD6] xor al,al ;AL löschen
[CC8] mov cl,01 ;einmal rotieren
[L58] cll ;Carry-Flag löschen
[D7D] rcl ah,cl ;AH-Register CL-mal links
rotieren
[33B] call warte ;Warterroutine aufrufen
[YDG] dec ch ;Rotationszähler um Eins
dekrementieren
[CUG] jnz loop8 ;wenn Zähler ungleich nul
l, dann wiederholen
[SUF] inc si ;Zeiger auf Datenspeicher
um Eins erhöhen
[N6A] pop dx ;Länge vom Stapel holen
[51B] dec dx ;und um Eins dekrementier
en
[BYF] jnz loop7 ;wenn Länge ungleich Null
, dann weiterer
[7RC] ;Durchgang; ansonsten ist
die Wiedergabe
[3S2] ;beendet
[6NE] mov al,0h ;sicherheitshalber Bit 0
ausschalten
[LUG] mov dx,prnadr ;dazu Adresse der Druck
erkarte in DX
[R89] out dx,al ;und AL an DX ausgeben
[H7D] pop bp ;Register wieder vom Stap
el holen
[LD8] ret ;Rücksprung zu Basic
[IS3] output2 endp

```

```

[000]
[MR2] ;-----
[AS5] ; Digitalisierung
[MR2] ;-----
[000]
[WG5] digitize proc far
[WM0] nop
[9E8] push bp ;Register sichern
[XY1] push ds
[C21] push ss
[YY1] push es
[GSB] mov dx,laenge ;Länge ins DX-Register
[D9K] mov di,pufadr ;Offset des Datenspeich
ers in den Zielindex
[1R7] loop3: push dx ;DX sichern
[EXA] mov ch,08h ;insgesamt 8mal rotieren
[0AH] mov dx,prnadr ;Basisadresse des Druck
erports in DX und
[5XF] inc dx ;daraus die Adresse des S
tatusregister
[U23] ;berechnen
[K3A] xor ah,ah ;alle Bits in AH löschen
[LPA] loop4: mov cl,01h ;einmal rotieren
[L58] cll ;Carry-Flag löschen
[D7D] rcl ah,cl ;AH-Register CL-mal links
rotieren
[CHB] in al,dx ;Statusregister lesen und
[VRB] test al,80h ;Bit 7 (Busy-Leitung) abf
ragen
[7HE] jz low1 ;wenn Null, dann nächsten
Befehl über-
[T03] or ah,01h ;springen
[MHB] ;Bit 0 des AH-Registers s
etzen
[SMC] low1: call warte ;Warterroutine aufrufen
[Z8D] nop ;Digitalisierung ist schn
eller als
[SK5] nop ;Wiedergabe
[WM0] nop
[W0E] dec ch ;Rotationszähler dekremen
tieren
[AHF] jnz loop4 ;wenn Zähler ungleich Nul
l, dann wieder-
[N42] ;holen
[VXG] mov [di],ah ;digitalisiertes Byte im
Datenspeicher
[TR2] ;ablegen
[9UF] inc di ;Zeiger auf Datenspeicher
um Eins erhöhen
[N6A] pop dx ;Länge vom Stapel holen
[51B] dec dx ;und um Eins dekrementier
en
[0LF] jnz loop3 ;wenn Länge <> Null, dann
weiterer Durchgang
[ZMD] ;ansonsten ist die Digita
lisierung beendet
[0BA] pop es ;Register restaurieren
[RN1] pop ss
[8N1] pop ds
[0A3] pop bp
[LD8] ret ;Rücksprung zu Basic
[000]
[MR2] ;-----
[DN5] ; Unterprogramm
[GN2] ; 'warte'
[MR2] ;-----
[K65] warte proc near
[CUS] push cx ;CX sichern
[7LG] mov cx,000eh ;Anzahl Warteschleifen-D
urchgänge in CX
[BKD] rep: nop ;3 Taktzyklen lang 'no op
eration'
[0Y8] loop rep ;CX-mal wiederholen
[E8A] pop cx ;CX wieder vom Stapel hol
[SPD] ret ;zurück zur aufrufenden R
outine
[5B5] warte endp
[000]
[YX4] ;-----
[4M2] ; Puffer
[FTA] ; für die digitalisierten Daten
[YX4] ;-----
[000]
[GE6] memory: db 7ef4h dup(?)
[000]
[LG4] digitize endp
[W99] code ends ;Ende des Segments
[4N7] end start ;Programmende

```

Listing 1. Diese Assembler-routinen sind in dem Data-Zeilen des Basic-Programms enthalten


```

[8F2] 10 'PC-Voice
[E3B] 20 'Sprachdigitalisierungs-System
[TD6] 30 '(C) 1988 by Martin Hepp
[B7A] 40 'Exklusiv für DOS International
[AT3] 50 'Version 2.0
[6P2] 60 ' 29.03.1988
[LC0] 70 '
[PFK] 80 'Data-Zeilen (enthalten die Maschinensprachero-
      utinen)
[NC0] 90 '
[BMB] 100 DATA &HEB, &H07, &H90, &HEB, &H34, &H90, &HEB
      , &H64, &H90, &H55
[H0C] 110 DATA &HBE, &HAC, &H80, &HBA, &HF4, &H7e, &H52
      , &HB5, &H08, &H8A
[EFB] 120 DATA &H24, &HE4, &H61, &H24, &HFD, &HF6, &HC4
      , &H80, &H74, &H02
[BHB] 130 DATA &HOC, &H02, &HE6, &H61, &HB1, &H01, &HF8
      , &HD2, &HD4, &HE8
[0SB] 140 DATA &H79, &H00, &HFE, &HCD, &H75, &HE7, &H46
      , &H5A, &H4A, &H75
[RUB] 150 DATA &HDD, &H24, &HFD, &HE6, &H61, &H5D, &HCB
      , &H55, &HBE, &HAC
[2XB] 160 DATA &H80, &HBA, &HF4, &H7e, &H52, &HB5, &H08
      , &H8A, &H24, &HBA
[1FB] 170 DATA &H78, &H03, &H90, &HF6, &HC4, &H80, &H74
      , &H02, &HB0, &H01
[KPB] 180 DATA &HEE, &H32, &HC0, &HB1, &H01, &HF8, &HD2
      , &HD4, &HE8, &H48
[BNB] 190 DATA &H00, &HFE, &HCD, &H75, &HE6, &H46, &H5A
      , &H4A, &H75, &HDC
[OMB] 200 DATA &HB0, &H00, &HBA, &H78, &H03, &HEE, &H5D
      , &HCB, &H90, &H55
[2YB] 210 DATA &H1E, &H16, &H06, &HBA, &HF4, &H7e, &HBF
      , &HAC, &H80, &H52
[5GB] 220 DATA &HB5, &H08, &HBA, &H78, &H03, &H42, &H32
      , &HE4, &HB1, &H01
[VPB] 230 DATA &HF8, &HD2, &HD4, &HEC, &HA8, &H80, &H74
      , &H03, &H80, &HCC
[SLB] 240 DATA &H01, &HE8, &H13, &H00, &H90, &H90, &H90
      , &HFE, &HCD, &H75
[7NB] 250 DATA &HE9, &H88, &H25, &H47, &H5A, &H4A, &H75
      , &HD9, &H07, &H17
[RNB] 260 DATA &H1F, &H5D, &HCB, &H51, &HB9, &H0E, &H00
      , &H90, &HE2, &HFD
[UA3] 270 DATA &H59, &HC3
[NL0] 280 '
[U5G] 300 'Maschinencode-Routinen in den Speicher schre-
      iben
[N4G] 310 CLEAR ,32767 'zuvor RAM für die Routin
      e und Ihre Daten
[WR6] 320 'reservieren
[C5C] 330 RESTORE 'Data-Zeiger zurücksetzte
      n
[2BF] 340 CHKSM=0:FOR A=32768! TO 32768!+171:READ B:POK
      E A,B:CHKSM=CHKSM+B:NEXT A
[DZK] 350 IF CHKSM<>22797 THEN BEEP:PRINT "Fehler in de
      n Data-Zeilen !!!":BEEP:PRINT "Programmausführung
      gestoppt":END
[MUH] 360 START=32768! 'Startadresse der Maschin
      encode-Routine
[63N] 370 OUTPUT=START 'Startadresse der Ausgabe
      routine 1 (PC-Lautsprecher)
[Z4M] 380 HIFI=START+3 'Startadresse der Ausgabe
      routine 2 (Hifi-Anlage)
[M2L] 390 DIGITIZE=START+6 'Startadresse der Digital
      isierungsroutine
[KDD] 400 ON ERROR GOTO 5260 'Fehlerbehandlung umleite
      n
[4KB] 410 WIDTH 80:CLS:KEY OFF 'Bildschirm löschen
[LEA] 420 GOSUB 5070 'Bildschirm umrahmen
[CR8] 430 LOCATE 6,1:T$="P C - V O I C E":GOSUB 5020
[TBR] 440 LOCATE 8,1:T$="Das DOS International Sprachdi
      gitalisierungs-System":GOSUB 5020
[SY8] 450 LOCATE 10,1:T$="Version 2.0":GOSUB 5020
[ZWB] 460 LOCATE 15,1:T$="Autor: Martin Hepp":GOSUB 502
      0
[7U2] 470 GOSUB 5160
[Y04] 500 'Hauptmenü
[YE2] 510 CLS:BEEP
[2K9] 520 GOSUB 5070 'Bildschirm umrahmen
[C6A] 530 LOCATE 5,7:PRINT "1 - Digitalisieren"
[L1E] 540 LOCATE 7,7:PRINT "2 - Ausgabe auf PC-Lautspre
      cher"
[BDD] 550 LOCATE 9,7:PRINT "3 - Ausgabe auf die Hifi-An
      lage"
[K98] 560 LOCATE 11,7:PRINT "4 - Speichern"
[DZ6] 570 LOCATE 13,7:PRINT "5 - Laden"
[SHA] 580 LOCATE 15,7:PRINT "6 - Parameter ändern"
[HB9] 590 LOCATE 20,3:PRINT "Ihre Wahl ";:INPUT W
[GA7] 600 IF W<1 OR W>6 THEN BEEP:GOTO 590
[UA7] 610 ON W GOTO 1000,2000,2500,3000,3500,3700
[7A5] 1000 'Digitalisieren
[EVO] 1010 '
[BX9] 1020 CLS:GOSUB 5070 'Rahmen aufbauen
[Z89] 1030 LOCATE 5,3:PRINT "Digitalisieren"
[N3D] 1040 LOCATE 9,7:PRINT "1 - Digitalisierung starte
      n"
[VGC] 1050 LOCATE 11,7:PRINT "9 - zurück ins Hauptmenü"

```

```

[8Y9] 1060 LOCATE 20,3:PRINT "Ihre Wahl ";:INPUT DW
[4M8] 1070 IF DW<>1 AND DW<>9 THEN BEEP:GOTO 1060
[6SC] 1080 IF DW=9 THEN GOTO 500 'zurück in Hauptm
      enü
[URU] 1090 CLS:T$="Bitte drücken Sie eine Taste, um die
      Digitalisierung zu starten":GOSUB 5020: PRINT
[RC6] 1100 IF INKEY$="" THEN GOTO 1100
[W6A] 1110 PRINT "Digitalisierung läuft..."
[7N3] 1120 CALL DIGITIZE
[L4C] 1130 BEEP:PRINT "Digitalisierung beendet"
[SZ2] 1140 GOSUB 5160
[3A2] 1150 GOTO 500
[LZA] 2000 'Wiedergabe über PC-Lautsprecher
[FV0] 2010 '
[XC9] 2020 CLS:GOSUB 5070 'Rahmen aufbauen
[34C] 2030 LOCATE 5,3:PRINT "Ausgabe auf PC-Lautspreche
      r"
[38B] 2040 LOCATE 9,7:PRINT "1 - Wiedergabe starten"
[WGC] 2050 LOCATE 11,7:PRINT "9 - zurück ins Hauptmenü"
[N0A] 2060 LOCATE 20,3:PRINT "Ihre Wahl ";:INPUT WW
[HP8] 2070 IF WW<>1 AND WW<>9 THEN BEEP:GOTO 2060
[P6D] 2080 IF WW=9 THEN GOTO 500 'zurück ins Haupt
      menü
[UTS] 2090 CLS:T$="Bitte drücken Sie eine Taste, um die
      Wiedergabe zu starten":GOSUB 5020: PRINT
[3G6] 2095 IF INKEY$="" THEN GOTO 2095
[Z28] 2100 PRINT "Wiedergabe läuft..."
[4M3] 2110 CALL OUTPUT
[2M9] 2120 BEEP:PRINT "Wiedergabe beendet"
[PZ2] 2130 GOSUB 5160
[LA2] 2140 GOTO 500
[KA4] 2500 'Wiedergabe über die Hifi-Anlage
[TV0] 2510 '
[PX9] 2520 CLS:GOSUB 5070 'Rahmen aufbauen
[DYB] 2530 LOCATE 5,3:PRINT "Ausgabe auf die Hifi-Anlag
      e"
[D8B] 2540 LOCATE 9,7:PRINT "1 - Wiedergabe starten"
[6HC] 2550 LOCATE 11,7:PRINT "9 - zurück ins Hauptmenü"
[3Z9] 2560 LOCATE 20,3:PRINT "Ihre Wahl ";:INPUT HW
[4N8] 2570 IF HW<>1 AND HW<>9 THEN BEEP:GOTO 2560
[H6D] 2580 IF HW=9 THEN GOTO 500 'zurück ins Haupt
      menü
[4US] 2590 CLS:T$="Bitte drücken Sie eine Taste, um die
      Wiedergabe zu starten":GOSUB 5020: PRINT
[BH6] 2595 IF INKEY$="" THEN GOTO 2595
[C28] 2600 PRINT "Wiedergabe läuft..."
[6W2] 2610 CALL HIFI
[CM9] 2620 BEEP:PRINT "Wiedergabe beendet"
[003] 2630 GOSUB 5160
[BA2] 2640 GOTO 500
[TA4] 3000 'Speichern
[GV0] 3010 '
[DX9] 3020 CLS:GOSUB 5070 'Rahmen aufbauen
[BH7] 3030 LOCATE 5,3:PRINT "Speichern"
[P6A] 3040 LOCATE 9,7:PRINT "1 - Daten speichern"
[VTC] 3045 LOCATE 11,7:PRINT "2 - Verzeichnis ausgeben"
[9HC] 3050 LOCATE 13,7:PRINT "9 - zurück ins Hauptmenü"
[60A] 3060 LOCATE 20,3:PRINT "Ihre Wahl ";:INPUT SW
[Y2B] 3070 IF SW<>1 AND SW<>2 AND SW<>9 THEN BEEP:GOTO
      3060
[6WE] 3072 IF SW=2 THEN GOSUB 5210:GOTO 3000 'Verzeichn
      is ausgeben
[L6D] 3080 IF SW=9 THEN GOTO 500 'zurück ins Haupt
      menü
[BPF] 3090 LOCATE 21,3:PRINT "Dateiname ";:INPUT D$:IF
      D$="" THEN BEEP:GOTO 3090
[L18] 3100 LNGE=PEEK(START+14)+PEEK(START+15)*256
[X8F] 3110 LOCATE 21,3:PRINT "Ich speichere Ihre Datei
      ...":LEFT$(D$,14);
[7N4] 3120 BSAVE D$,32768!,LNGE
[0E3] 3130 BEEP:GOTO 500
[302] 3500 'Laden
[VH9] 3510 CLS:GOSUB 5070 'Rahmen aufbauen
[5U5] 3520 LOCATE 5,3:PRINT "Laden"
[E99] 3530 LOCATE 9,7:PRINT "1 - Datei laden"
[HTC] 3540 LOCATE 11,7:PRINT "2 - Verzeichnis ausgeben"
[LHC] 3550 LOCATE 13,7:PRINT "9 - zurück ins Hauptmenü"
[MZ9] 3560 LOCATE 20,3:PRINT "Ihre Wahl ";:INPUT LW
[S1B] 3570 IF LW<>1 AND LW<>2 AND LW<>9 THEN BEEP:GOTO
      3560
[XWE] 3572 IF LW=2 THEN GOSUB 5210:GOTO 3500 'Verzeichn
      is ausgeben
[P6D] 3580 IF LW=9 THEN GOTO 500 'zurück ins Haupt
      menü
[NPF] 3590 LOCATE 21,3:PRINT "Dateiname ";:INPUT D$:IF
      D$="" THEN BEEP:GOTO 3590
[ZAC] 3610 LOCATE 21,3:PRINT "Ich lade Datei: ";LEFT$(D$
      ,14);
[YA3] 3620 BLOAD D$,32768!
[9A2] 3630 GOTO 500
[8L6] 3700 'Parameter ändern
[VY0] 3710 '
[55A] 3720 CLS:GOSUB 5070 'Rahmen erstellen
[S8D] 3730 LOCATE 5,3:PRINT "Parameter anzeigen/ändern"
[5MC] 3740 LOCATE 9,7:PRINT "1 - Länge der Aufzeichnung
      "
[8LA] 3750 LOCATE 11,7:PRINT "2 - Geschwindigkeit"
[W1A] 3760 LOCATE 13,7:PRINT "3 - Druckerkarte"

```



```

[9KC] 3770 LOCATE 15,7:PRINT "9 - zurück ins Hauptmenü"
[E0A] 3780 LOCATE 20,3:PRINT "Ihre Wahl ";:INPUT PW
[4VA] 3790 IF (PW <1 OR PW>3) AND PW<>9 THEN BEEP:GOTO
3780
[VYD] 3800 IF PW=1 THEN GOTO 4100 'Länge der Aufzei
chnung
[MPB] 3810 IF PW=2 THEN GOTO 4300 'Geschwindigkeit
[5CA] 3820 IF PW=3 THEN GOTO 4500 'Druckerkarte
[DA2] 3830 GOTO 500
[64D] 4100 'Länge der Aufzeichnung anzeigen/ändern
[LVO] 4110 '
[U4A] 4120 CLS:GOSUB 5070 'Rahmen erstellen
[DPB] 4130 LOCATE 5,3:PRINT "Länge der Aufzeichnung"
[SKA] 4140 LOCATE 9,7:PRINT "1 - Länge verändern"
[GPD] 4150 LOCATE 11,7:PRINT "8 - zurück ins Parametern
enü"
[FHC] 4160 LOCATE 13,7:PRINT "9 - zurück ins Hauptmenü"
[6GB] 4165 LOCATE 17,7:PRINT "Länge: ";(PEEK(START+14))+
256*PEEK(START+15))
[5Z9] 4170 LOCATE 20,3:PRINT "Ihre Wahl ";:INPUT WL
[R1B] 4180 IF WL<>1 AND WL<>8 AND WL<>9 THEN BEEP:GOTO
4170
[APE] 4190 IF WL=8 THEN GOTO 3700 'zurück ins Parameter
menü
[ARC] 4200 IF WL=9 THEN GOTO 500 'zurück ins Hauptmenü
[H7C] 4210 LOCATE 21,3:PRINT "Neuer Wert (10 - 32500) "
;:INPUT NL
[L09] 4220 IF NL <1 OR NL>32500 THEN BEEP:GOTO 4210
[0S7] 4225 NWH=INT(NL/256):NWL=NL-NWH*256
[669] 4230 POKE (START+14),NWL:POKE (START+15),NWH
[X69] 4240 POKE (START+62),NWL:POKE (START+63),NWH
[2Y9] 4250 POKE (START+114),NWL:POKE (START+115),NWH
[1M2] 4260 GOTO 3700
[ESA] 4300 'Geschwindigkeit anzeigen/ändern
[RVO] 4310 '
[Y4A] 4320 CLS:GOSUB 5070 'Rahmen erstellen
[91A] 4330 LOCATE 5,3:PRINT "Geschwindigkeit"
[B2E] 4340 LOCATE 9,7:PRINT "1 - Geschwindigkeit veränd
ern"
[MPD] 4350 LOCATE 11,7:PRINT "8 - zurück ins Parametern
enü"
[LHC] 4360 LOCATE 13,7:PRINT "9 - zurück ins Hauptmenü"
[9SE] 4365 LOCATE 17,7:PRINT "Geschwindigkeit: ";(PEEK(
START+165))
[0Z9] 4370 LOCATE 20,3:PRINT "Ihre Wahl ";:INPUT GW
[N1B] 4380 IF GW<>1 AND GW<>8 AND GW<>9 THEN BEEP:GOTO
4370
[MFE] 4390 IF GW=8 THEN GOTO 3700 'zurück ins Parameter
menü
[MRC] 4400 IF GW=9 THEN GOTO 500 'zurück ins Hauptmenü
[R3H] 4410 LOCATE 21,3:PRINT "Neuer Wert (1-schnell, 25
5-langsam) ";:INPUT NW
[LG8] 4420 IF NW <1 OR NW>255 THEN BEEP:GOTO 4410
[925] 4430 POKE (START+165),NW
[ZL2] 4440 GOTO 3700
[YOE] 4500 'Adresse der Druckerkarte anzeigen/ändern
[VVO] 4510 '
[25A] 4520 CLS:GOSUB 5070 'Rahmen erstellen
[3B8] 4530 LOCATE 5,3:PRINT "Druckerkarte"
[U3D] 4540 LOCATE 9,7:PRINT "1 - Basisadresse verändern
"
[SPD] 4550 LOCATE 11,7:PRINT "8 - zurück ins Parametern
enü"
[RHC] 4560 LOCATE 13,7:PRINT "9 - zurück ins Hauptmenü"
[4NG] 4565 LOCATE 17,7:PRINT "Basisadresse: ";(PEEK(STA
RT+70))+256*PEEK(START+71))
[80A] 4570 LOCATE 20,3:PRINT "Ihre Wahl ";:INPUT PW
[G4B] 4580 IF PW<>1 AND PW<>8 AND PW<>9 THEN BEEP:GOTO
4570
[1GE] 4590 IF PW=8 THEN GOTO 3700 'zurück ins Parameter
menü

```

```

[1SC] 4600 IF PW=9 THEN GOTO 500 'zurück ins Hauptmenü
[M7L] 4610 LOCATE 21,3:PRINT "Neuer Wert (956 [$3BC], 8
88 [$378] oder 632 [$278]) ";:INPUT NA
[MWB] 4620 IF NA<>956 AND NA<>888 AND NA<>632 THEN BEEP
:GOTO 4610
[277] 4625 NAH=INT(NA/256):NAL=NA-NAH*256
[U28] 4630 POKE (START+70),NAL:POKE (START+71),NAH
[HU9] 4640 POKE (START+103),NAL:POKE (START+104),NAH
[UU9] 4650 POKE (START+123),NAL:POKE (START+124),NAH
[9M2] 4660 GOTO 3700
[R25] 5000 'Hilfsroutinen
[KV0] 5010 '
[5D8] 5020 'Text zentriert ausgeben
[YKC] 5030 'T$ enthält den auszugebenden String
[ENE] 5040 PS=(80-LEN(T$))/2 'Anzahl Leerzeichen bere
chnen
[SKX] 5050 LOCATE ,PS:PRINT T$;'Cursor positionieren, d
ann Textstring
[922] 5060 RETURN
[Y67] 5070 'Bildschirm umrahmen
[W70] 5080 '
[OWC] 5090 LOCATE 1,1 'Cursor --> Bildschirm an
fang
[BHF] 5100 PRINT CHR$(201);:FOR C=1 TO 78:PRINT CHR$(20
5);:NEXT C:PRINT CHR$(187);
[D4K] 5110 LOCATE 2,1:PRINT CHR$(186);:T$="P C - V O I
C E":GOSUB 5020:LOCATE 2,80:PRINT CHR$(186);
[GGH] 5120 LOCATE 3,1:PRINT CHR$(204);:FOR F=1 TO 78:PR
INT CHR$(205);:NEXT F:PRINT CHR$(185);
[5GH] 5130 FOR D=4 TO 23:LOCATE D,1:PRINT CHR$(186):LOC
ATE D,80:PRINT CHR$(186);:NEXT D
[2KF] 5140 PRINT CHR$(200);:FOR E=1 TO 78:PRINT CHR$(20
5);:NEXT E:PRINT CHR$(188);
[822] 5150 RETURN
[PW7] 5160 'auf Tastendruck warten
[6UD] 5170 LOCATE 25,1:T$="Taste drücken für weiter":GO
SUB 5020
[ZF6] 5180 IF INKEY$="" THEN GOTO 5180
[U96] 5190 LOCATE 25,1:PRINT SPC(80);
[V12] 5200 RETURN
[1E7] 5210 'Verzeichnis ausgeben
[TV0] 5220 '
[M23] 5230 CLS:FILES
[Y22] 5240 GOSUB 5160
[A22] 5250 RETURN
[4D6] 5260 'Fehlerbehandlung
[8W0] 5270 '
[TN4] 5280 'File not found
[L3N] 5290 IF ERR=53 THEN LOCATE 22,1:T$="Datei nicht g
efunden":GOSUB 5020:GOSUB 5160:RESUME 500
[R24] 5300 'Disk full
[VYN] 5310 IF ERR=61 THEN LOCATE 22,1:T$="Diskette/Fest
platte voll":GOSUB 5020:GOSUB 5160:RESUME 500
[366] 5320 'Disk media error
[F3L] 5330 IF ERR=72 THEN LOCATE 22,1:T$="Diskettenfehl
er":GOSUB 5020:GOSUB 5160:RESUME 500
[SN4] 5340 'Disk not ready
[74N] 5350 IF ERR=71 THEN LOCATE 22,1:T$="Laufwerk nich
t bereit":GOSUB 5020:GOSUB 5160:RESUME 500
[UA7] 5360 'Disk write protected
[YXP] 5370 IF ERR=70 THEN LOCATE 22,1:T$="Diskette schr
eibgeschützt":GOSUB 5020:GOSUB 5160:RESUME 500
[TB4] 5380 'bad file name
[VUN] 5390 IF ERR=64 THEN LOCATE 22,1:T$="Unzulässiger
Dateiname":GOSUB 5020:GOSUB 5160:RESUME 500
[CWH] 5400 ON ERROR GOTO 0 'bei allen andere
n Fehlern --> Programmende
[000]

```

Listing 2. Die Betriebssoftware von PC-Voice ist in GW-Basic ge-
schrieben; die Digitalisierung erledigt eine Assembleroutine

Eine Bitte an unsere Abonnenten

Vermerken Sie bei Schriftverkehr und Zahlungen neben der
vollständigen Anschrift stets Ihre Abo-Nummer.

Sie vermeiden damit unnötige Verzögerungen bei der Bearbeitung
Ihres Abonnements.

Vielen Dank

Ihre DMV-Versandabteilung

Soundprogrammierung in Basic und Pascal

Mit diesen Programmen haben Sie zwar keinen Steinway-Flügel im Arbeitszimmer stehen, aber wenn Sie schnell eine Melodie ausprobieren wollen, stehen Ihnen mit »bmaster.bas« interessante Funktionen zur Verfügung.

Nachdem Sie das Programm (Listing 1) mit »run« gestartet haben, erscheint die Frage, ob Sie eine Informationskarte ausdrucken wollen. In dieser Informationskarte ist die Tastaturbelegung aufgelistet. Nachdem Sie die Frage beantwortet haben, geben Sie die Abspielgeschwindigkeit für die Pads an. Hier können Sie zwischen Werten von 1 bis 64 wählen; dann gelangen Sie in den eigentlichen Spielmodus.

Bei den Pads können Sie verschiedene Oktaven einstellen

Die erste Unterteilung der Tastatur betrifft die Pads. Hier stehen Ihnen mehrere Funktionen zur Verfügung. Sie bestehen aus den Funktionstasten »F1« bis »F8«, der Tabulatortaste, der Entertaste und der Taste mit dem Minuszeichen. In diesem Spielmodus lassen sich die Oktaven mit den Tasten »<« und »>« regeln. Die Geschwindigkeit variieren Sie mit den Funktionstasten »F9« und »F10«. Wenn Sie ein Staccato spielen wollen, drücken Sie die Punktaste ».«, und für Legato ist die Taste mit dem Komma vorgesehen. Die Pads können Sie mit 64 verschiedenen Geschwindigkeiten, mit sieben Oktaven, Staccato und Legato spielen. Mit den Pads lassen sich dadurch interessante Effekte erzielen. Probieren Sie es einfach aus!

Vertrauter wird Ihnen die Art der zweiten Tastaturbelegung sein. Die untere Tastenreihe, von »Y« bis »M«, die darüberliegende Reihe mit den Tasten »A«, »S«, »F«, »G«, »H«, wieder die Tasten darüber von »Q« bis »U«, sowie die Zahlen von 1 bis 5 simulieren eine Klaviertastatur. Dazu gehören noch die Tasten »I«, »O«, »K« und »L«, mit denen Sie auch noch die Tonlänge zum direkten Spielen manipulieren. Geben Sie für die Länge den Wert »1« ein, so bewirkt dies ein kurzes Anschlagen des Tones, während die Länge 16 einen Ton voll ausspielt. Länge 16 ist übrigens voreingestellt und empfiehlt sich deshalb, weil man mitten im Ton zu einem anderen wechseln kann. Wenn Sie die Leertaste betätigen, bricht der gerade gespielte Ton einfach ab.

Sie können auch die Tonlänge beim direkten Spielen manipulieren

Die Enter-Taste aktiviert eine Funktion, die es Ihnen erlaubt, einen Vierertakt anzuschlagen. Betätigen Sie die Taste zu schnell, gleicht das Programm die Unregelmäßigkeit durch Viertelpausen aus. Ein Druck auf die Escape-Taste aktiviert das Glissando.

Wollen Sie ein heißes Solo spielen, empfiehlt sich die folgende Einstellung: Geschwindigkeit: 32/44 Oktave: 4 Sie könne natürlich auch mit anderen Einstellungen experimentieren, die Ihrem persönlichen Geschmack näher kommen.

Die Farbpalette ist so angepaßt, daß sich sowohl im CGA- als auch im Herculesmodus die Farben klar abheben.

Das kleinen Turbo-Pascal-Programm (Listing 2) simuliert ebenfalls ein Klavier, bei dem Sie den Tonumfang noch erweitern können. Beethoven läßt grüßen...

(Jochen Schmidt/gh)

```
[1F9] 100 COLOR 7,0:CLS:SCREEN 0:WIDTH 80:KEY OFF
[AZ4] 101 LOCATE 12,17:COLOR 7,0
[S2H] 105 PRINT "Wollen Sie eine Informationskarte drucken (j/n)"
[YM5] 110 C$=INKEY$:IF C$="" THEN 110
[3U5] 115 IF C$="j" OR C$="J" THEN 200
[8W5] 120 IF C$="n" OR C$="N" THEN 1000
[LSB] 200 LOCATE 14,30:PRINT "<SPACE> um anzufangen..."
[9A7] 205 B$=INKEY$:IF B$<>CHR$(32) THEN 205
[N34] 207 ON ERROR GOTO 600
[XB8] 295 LPRINT "*****"
[LU8] 300 LPRINT "      * B E A T M A S T"
[CB8] 305 LPRINT "*****"
[522] 306 LPRINT
[1TB] 310 LPRINT " Space           Beenden der Note"
[ESB] 315 LPRINT " Enter           Rhythmuspad"
[9EA] 320 LPRINT " Esc           Glissando"
[2NB] 325 LPRINT " Backspace      Exit"
[AD9] 330 LPRINT " ,             Legato"
[ZW9] 335 LPRINT " .             Staccato"
[UW8] 337 LPRINT " -             Pad"
[WS9] 340 LPRINT " F1-F8          Pads"
[EGD] 345 LPRINT " F9-F10         Geschwindigkeit für F1-F8"
[CU9] 350 LPRINT " I             Länge 1"
```

```
[EW9] 355 LPRINT " O           Länge 4"
[HV9] 360 LPRINT " K           Länge 8"
[AY9] 365 LPRINT " L           Länge 16"
[VHB] 370 LPRINT " <          Oktave runter"
[MZA] 375 LPRINT " >          Oktave rauf"
[2TC] 380 LPRINT " Y/X/C/V/B/N/M   Noten c/d/e/f/g/a/b"
[1VB] 385 LPRINT " A/S/F/G/H       Noten c#/d#/f#/g#/a#"
[Z3C] 390 LPRINT " Q/W/E/R/T/Z/U   Noten C/D/E/F/G/A/B"
[4YA] 395 LPRINT " 1/2/4/5/6       Noten C#/D#/F#/G#/A#"
[19B] 400 LPRINT " Tabulator       Pad"
[GF2] 500 GOTO 1000
[MMH] 600 CLS:LOCATE 12,30:PRINT "Es ist kein Drucker angeschlossen..."
[0KC] 605 LOCATE 16,35:PRINT "Nochmal versuchen (j/n)"
[682] 610 GOTO 110
[E1A] 1000 COLOR 7,0:CLS:L$="1":R$="ms":X=130.81:PLAY"o3"
[VG9] 1055 LOCATE 20,30:PRINT "Vorschlag:32/44"
[AF4] 1060 COLOR 0,7:S$="132"
[E23] 1065 LOCATE 8,20
[VBC] 1070 INPUT "Geschwindigkeit für F9 (1-64):",EU$
[XH2] 1080 U$=L$+EU$
[V23] 1085 LOCATE 12,20
[0EC] 1090 INPUT "Geschwindigkeit für F10(1-64) :",EV$
[KH2] 1100 V$=L$+EV$
[GA6] 1127 COLOR 7,0:CLS:N=16:S$="132"
```



```

[X13] 1129 LOCATE 2,30
[6R6] 1130 PRINT "B E A T M A S T E R"
[VN7] 1180 LOCATE 10,3:PRINT "F1 =pad 1"
[AP7] 1190 LOCATE 11,3:PRINT "F2 =pad 2"
[XN7] 1200 LOCATE 12,3:PRINT "F3 =pad 3"
[CP7] 1210 LOCATE 13,3:PRINT "F4 =pad 4"
[VP7] 1220 LOCATE 14,3:PRINT "F5 =pad 5"
[AR7] 1230 LOCATE 15,3:PRINT "F6 =pad 6"
[TR7] 1240 LOCATE 16,3:PRINT "F7 =pad 7"
[8S7] 1250 LOCATE 17,3:PRINT "F8 =pad 8"
[249] 1260 LOCATE 18,3:PRINT "F9 =Geschw. "+EU$+" "
[K59] 1270 LOCATE 19,3:PRINT "F10=Geschw. "+EV$+" "
[UX4] 1275 COLOR 0,7:LOCATE 13,42
[XEA] 1280 PRINT " Q W E R T Z U Oktave ";CHR$(24);" "
[L33] 1285 LOCATE 15,42
[UN9] 1290 PRINT " A S F G H Halb ";CHR$(24);" "
[243] 1295 LOCATE 17,42
[CH8] 1300 PRINT " Y X C V B N M normal "
[WM4] 1302 COLOR 0,7:LOCATE 11,42
[YE9] 1305 PRINT " 1 2 4 5 6 Halb ";CHR$(24);" "
[NA8] 1310 COLOR 10,0:LOCATE 19,57:PRINT "by"
[2SA] 1320 LOCATE 21,52:PRINT "Jochen Schmidt"
[ST8] 1360 LOCATE 20,47:COLOR 24,0:PRINT CHR$(14)
[PT8] 1370 LOCATE 20,72:COLOR 24,0:PRINT CHR$(14)
[S35] 1380 LOCATE 23,1:COLOR 14,0
[MV9] 1400 LOCATE 10,20:PRINT "Esc=Glissando"
[NY7] 1405 LOCATE 11,20:PRINT "Space=cut"
[UKA] 1410 LOCATE 12,20:PRINT "Backspace=ende"
[XKA] 1415 LOCATE 13,20:PRINT "Enter=Glissando"
[H48] 1420 LOCATE 14,20:PRINT "I=Länge 1"
[068] 1425 LOCATE 15,20:PRINT "O=Länge 4"
[T58] 1430 LOCATE 16,20:PRINT "K=Länge 8"
[UD8] 1435 LOCATE 17,20:PRINT "L=Länge 16"
[6X9] 1440 LOCATE 18,20:PRINT "=<Oktave ";CHR$(25)
[1Y9] 1445 LOCATE 19,20:PRINT "=>Oktave ";CHR$(24)
[EZ9] 1450 LOCATE 20,20:PRINT "Tabulator=Pad "
[U36] 1500 LOCATE 8,1:PRINT CHR$(201)
[GB6] 1501 LOCATE 8,80:PRINT CHR$(187)
[M66] 1503 LOCATE 23,1:PRINT CHR$(200)
[FF7] 1504 LOCATE 23,80:PRINT CHR$(188):X=2
[GF3] 1510 FOR I=1 TO 78
[ED6] 1512 LOCATE 8,X:PRINT CHR$(205)
[286] 1515 LOCATE 23,X:PRINT CHR$(205)
[ER1] 1516 X=X+1
[WML] 1520 NEXT
[NK1] 1530 Y=9
[BF3] 1540 FOR I=1 TO 14
[NB6] 1542 LOCATE Y,1:PRINT CHR$(186)
[AH6] 1545 LOCATE Y,80:PRINT CHR$(186)
[8R1] 1550 Y=Y+1
[8N1] 1560 NEXT
[813] 1562 LOCATE 22,3
[RB9] 1855 PRINT " ,=Legato / . =Staccato / --Pad"
[CT2] 1860 KEY(1) ON
[HT2] 1870 KEY(2) ON
[PT2] 1880 KEY(3) ON
[VT2] 1890 KEY(4) ON
[8Z2] 1900 KEY(5) ON
[EZ2] 1910 KEY(6) ON
[MZ2] 1920 KEY(7) ON
[UZ2] 1930 KEY(8) ON
[003] 1940 KEY(9) ON
[M13] 1950 KEY(10) ON
[RS4] 1960 FOR I=1 TO 100000000#
[3V5] 1970 AS=INKEY$:IF AS="" THEN 1970
[PK4] 1980 ON KEY(1) GOSUB 2480
[YK4] 1990 ON KEY(2) GOSUB 2490
[YH4] 2000 ON KEY(3) GOSUB 2500
[6K4] 2010 ON KEY(4) GOSUB 2510
[EK4] 2020 ON KEY(5) GOSUB 2520
[PK4] 2030 ON KEY(6) GOSUB 2530
[YK4] 2040 ON KEY(7) GOSUB 2540
[6L4] 2050 ON KEY(8) GOSUB 2550
[EL4] 2060 ON KEY(9) GOSUB 2560
[KN4] 2070 ON KEY(10) GOSUB 2570
[286] 2080 IF AS="y" THEN GOSUB 11000
[486] 2090 IF AS="x" THEN GOSUB 12000
[646] 2100 IF AS="c" THEN GOSUB 13000
[C76] 2110 IF AS="v" THEN GOSUB 14000
[F46] 2120 IF AS="b" THEN GOSUB 15000
[K66] 2130 IF AS="n" THEN GOSUB 16000
[M66] 2140 IF AS="m" THEN GOSUB 17000
[XW5] 2150 IF AS="1" THEN GOSUB 32000
[9X5] 2160 IF AS="2" THEN GOSUB 33000
[1Y5] 2180 IF AS="4" THEN GOSUB 35000
[DY5] 2190 IF AS="5" THEN GOSUB 36000
[XX5] 2200 IF AS="6" THEN GOSUB 37000
[S46] 2220 IF AS="a" THEN GOSUB 18000
[T76] 2230 IF AS="s" THEN GOSUB 19000
[156] 2250 IF AS="f" THEN GOSUB 21000
[D56] 2260 IF AS="g" THEN GOSUB 22000
[T56] 2270 IF AS="h" THEN GOSUB 23000
[M76] 2290 IF AS="q" THEN GOSUB 25000

```

```

[X76] 2300 IF AS="w" THEN GOSUB 26000
[A56] 2310 IF AS="e" THEN GOSUB 27000
[K76] 2320 IF AS="r" THEN GOSUB 28000
[386] 2330 IF AS="t" THEN GOSUB 29000
[386] 2340 IF AS="z" THEN GOSUB 30000
[H76] 2350 IF AS="u" THEN GOSUB 31000
[VX6] 2360 IF AS=CHR$(13) THEN GOSUB 6000
[LT6] 2370 IF AS=CHR$(32) THEN SOUND X,0
[8K4] 2380 IF AS="i" THEN N=1
[6M4] 2390 IF AS="k" THEN N=8
[PM4] 2400 IF AS="." THEN RS="ms"
[8L4] 2410 IF AS="o" THEN N=4
[7K4] 2420 IF AS="l" THEN N=16
[7M4] 2430 IF AS="," THEN RS="ml"
[HT4] 2440 IF AS="<" THEN PLAY "<"
[2U4] 2450 IF AS=">" THEN PLAY ">"
[2KF] 2452 IF AS="-" THEN PLAY "ms 164 ff0fp64dd0dp64cc0
cp64dd0d000ee0eccc"
[R47] 2455 IF AS=CHR$(27) THEN GOSUB 40000
[7F5] 2460 IF AS=CHR$(8) THEN END
[WWA] 2465 IF AS=CHR$(9) THEN PLAY "ms 144 o3 c00c0cc"
[AN1] 2470 NEXT
[UN2] 2475 GOTO 1960
[027] 2480 PLAY RS+SS+"d00f00g00f":RETURN
[X37] 2490 PLAY RS+SS+"f00g00eg0d":RETURN
[VX6] 2500 PLAY RS+SS+"c00e00g0f":RETURN
[277] 2510 PLAY RS+SS+"f00f0e0d":RETURN
[M87] 2520 PLAY RS+SS+"a0add0a":RETURN
[N37] 2530 PLAY RS+SS+"d00fg0e":RETURN
[LB7] 2540 PLAY RS+SS+"c00f0e0fe":RETURN
[R97] 2550 PLAY RS+SS+"da0e0f":RETURN
[XE3] 2560 SS=US:RETURN
[1F3] 2570 SS=V$:RETURN
[SX5] 6000 PLAY "ms 132 cp4cp4"
[C22] 6005 RETURN
[HD8] 11000 SOUND X,0:X=130.81:SOUND X,N:RETURN
[NE8] 12000 SOUND X,0:X=146.83:SOUND X,N:RETURN
[NE8] 13000 SOUND X,0:X=164.81:SOUND X,N:RETURN
[DE8] 14000 SOUND X,0:X=174.61:SOUND X,N:RETURN
[BB8] 15000 SOUND X,0:X=196.1:SOUND X,N:RETURN
[P98] 16000 SOUND X,0:X=220.1:SOUND X,N:RETURN
[9F8] 17000 SOUND X,0:X=246.94:SOUND X,N:RETURN
[7F8] 18000 SOUND X,0:X=138.81:SOUND X,N:RETURN
[2F8] 19000 SOUND X,0:X=155.83:SOUND X,N:RETURN
[HE8] 21000 SOUND X,0:X=185.61:SOUND X,N:RETURN
[FA8] 22000 SOUND X,0:X=208.1:SOUND X,N:RETURN
[4A8] 23000 SOUND X,0:X=233.1:SOUND X,N:RETURN
[0E8] 25000 SOUND X,0:X=261.63:SOUND X,N:RETURN
[SE8] 26000 SOUND X,0:X=293.66:SOUND X,N:RETURN
[Z8] 27000 SOUND X,0:X=329.63:SOUND X,N:RETURN
[DE8] 28000 SOUND X,0:X=349.23:SOUND X,N:RETURN
[6B8] 29000 SOUND X,0:X=392.1:SOUND X,N:RETURN
[U98] 30000 SOUND X,0:X=440.1:SOUND X,N:RETURN
[3F8] 31000 SOUND X,0:X=493.88:SOUND X,N:RETURN
[7B8] 32000 SOUND X,0:X=277.1:SOUND X,N:RETURN
[P98] 33000 SOUND X,0:X=311.1:SOUND X,N:RETURN
[7E8] 35000 SOUND X,0:X=380.63:SOUND X,N:RETURN
[EE8] 36000 SOUND X,0:X=416.66:SOUND X,N:RETURN
[4F8] 37000 SOUND X,0:X=466.63:SOUND X,N:RETURN
[4X4] 40000 FOR I=300 TO 310 STEP 1
[8F5] 41000 SOUND I,13:NEXT:RETURN
[WD1] 50000 END

```

Listing 1. Mit »bmaster.bas« steht Ihnen eine Fülle von Klängen zur Verfügung

```

[5W4] 0: program musik;
[000] 1:
[MCC] 2: const C = 261; D = 293; E = 329; F = 349; G
= 392; H = 493; CC = 523;
[ES6] 3: DD = 587; FF = 698; A = 440; EE = 659
;
[PVB] 4: CIS = 277; DIS = 311; FIS = 370; GIS
= 415; AIS = 466; CCIS = 554;
[OK4] 5: DDIS = 622; FFIS = 740;
[000] 6:
[MR3] 7: var x: char;
[F04] 8: t, i: integer;
[000] 9:
[H95] 10: procedure bild;
[BH1] 11: begin
[FTD] 12: { = 219, = 218, = 196, = 1
94, = 193, = 217, = 179}
[BX3] 13: { = 191, = 192}
[2Y5] 14: writeln; writeln;

```


PC-Spielebox No. 1

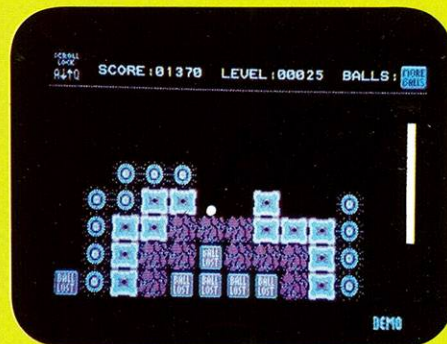
BIMBO II

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 Staße bis zu ihrem Ende zu fahren?

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

Best.-Nr. 129

Lauffähig auf allen kompatiblen PC's mit EGA- oder CGA-Farbgrafikkarte unter MS-DOS, Steuerung per Tastatur, bei den AMSTRAD PC's auch per Maus.

Beide Spiele auf einer Diskette zum günstigen Preis von

49, – DM

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 vollkommen **menuegesteuert**!
- besitzt eine **Supergrafik**!
- stellt zu jeder Frage wahlweise eine oder fünf Antworten 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 mit einem Vielfachen davon!
- 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.

Best.-Nr. 161

49, – DM*

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.

unverbindl. Preisempf.

Bestellen Sie noch heute mittels unserer Postkarte

DMV GmbH · Abt. Software · Postfach 250 · 3440 Eschwege

Star-Planer^{PC}



Schwupp-di-wupp

Das deutsche,
professionelle
Kombiprogramm
für Tabellen-
kalkulation / Daten-
bank / Grafik zum
Überraschungspreis!

F Ü R I B M K O M P A T I B L E C O M P U T E R

Star-Planer PC läuft unter PC-DOS / MS-DOS 2.11 (aufwärtskompatibel) auf jedem IBM-kompatiblen PC ab 256 kB RAM in Farbe und monochrom. Seinen hohen Komfort für den Anwender entwickelt Star-Planer PC bei vollem Speicherausbau und mit einem Grafiktreiber — Hercules, EGA oder CGA.

Vergleichen Sie gern die Leistung von Star-Planer PC auf Ihrem privaten System mit dem besten Programm, das an Ihrem beruflichen Arbeitsplatz auf einem Professional-System verfügbar ist.

Tabellenkalkulation

Mit Star-Planer PC können Sie auch zuhause und unterwegs problemlos auf die Daten vieler anderer Programme zugreifen und sie weiterverarbeiten; denn Star-Planer PC ist Files- und Makro-kompatibel zu Lotus 1-2-3, Vers. 1.1A - dem Standard in Sachen Tabellenkalkulation. Damit gehen Sie auf Nummer Sicher! Und kompatibel heißt für STAR-DIVISION - die Macher von Star-Planer PC und anderen Software-Delikatessen - auf deutsch „zusammenpassend / vereinbar“.

Star-Planer PC besitzt den Standard internationaler Spitzenprodukte — einfach top. Star-Planer PC ist die deutsche Tabellenkalkulation mit rechnender Grafik-Simultan-Anzeige und Datenbank.

Grafik

Alles*), was Sie über und mit professioneller Tabellenkalkulation gelernt haben, können Sie bei Star-Planer PC anwenden: Das Arbeitsblatt eröffnen, einfügen, löschen, radieren, benennen, fensterln, formatieren, justieren, neu-berechnen, kopieren, versetzen, transferieren, grafisch aufbereiten, verwalten, suchen, finden, analysieren, zurücksetzen. . . Alles, was Ihnen fremd geblieben ist, wird mit der deutschen Benutzerführung und einem phantastisch guten Handbuch in deutscher Sprache zur schnellen Routine.

Datenbank

*) Fast alles! Die Faxen mit Kopierschutz und Original-Startdiskette dürfen Sie getrost vergessen. Sie starten ab Floppy oder Festplatte — schwupp-di-wupp!

Star-Planer PC ist das Tabellenkalkulationsprogramm für Leute, die auch dann rechnen können, wenn's um den Anschaffungspreis für Spitzensoftware geht. Und für die Preisdifferenz zwischen Star-Planer PC und in der Leistung vergleichbarer Software erwerben sie eine Festplatte zum superschnellen Arbeiten — oder fliegen in den fälligen Urlaub. Aber verraten Sie Ihrem Chef nicht, für welchen Preis Sie Star-Planer PC erworben haben, falls er Ihnen die Heimarbeitsstunden vergütet. . .

Raubkopieren lohnt nicht — denn Star-Planer PC kostet Sie lediglich ein Lächeln und nur

DM 298,-*

* Unverbindliche Preisempfehlung



I N F O - C O U P O N



Star-Planer

Logo — ich will mehr: Informationen über Star-Planer PC und die weiteren PC-Software-Angebote von  STAR-DIVISION

Name, Vorname

Straße, Nr.

PLZ, Ort

Bitte ausschneiden, auf eine Postkarte kleben oder einen Briefumschlag stecken und an STAR-DIVISION GmbH, Postfach 2830 in 2120 Lüneburg abschicken

STAR-DIVISION

POSTFACH 2830 • 2120 Lüneburg • Tel.: (041 31) 70090

Hinweis: Lotus 1-2-3 ist eingetragenes Warenzeichen der Lotus-Development Corp., USA; MS-DOS der Microsoft Corporation, USA; PC-DOS der IBM Corp., USA.