DOS EXTRA Nr. 14 – 1991

DOS EXTRA

18,– DM 145 öS 18,– sfr 55,– Fmk

Sonderausgabe DOS EXTRA

Nr. 14 1991

ISSN 0934 – 2842

DMV-Verlag

Eine Publikation der
DOS INTERNATIONAL

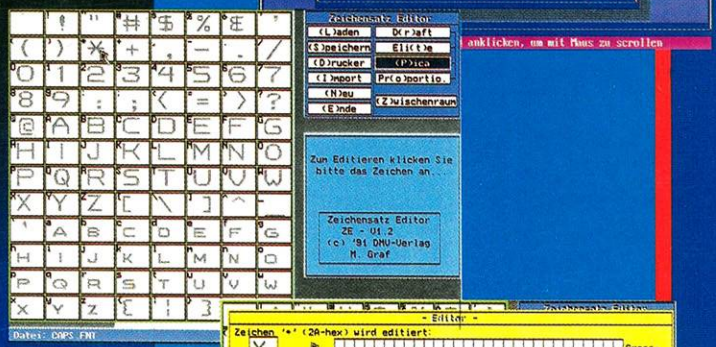
SBACKUP

Datensicherung professionell



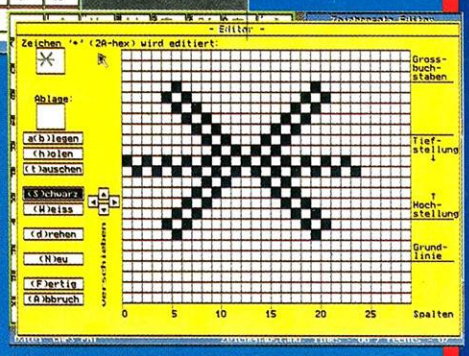
GRUNDWISSEN

Strichcode entschlüsseln
C-Makro-Programmierung



ZUM ABTIPPEN

Quelltextoptimierung
3D-Animation
So verwalten Sie
TSR-Programme



TIPS, TRICKS & UTILITIES



Professionelle Grafik-Designs auf dem PC









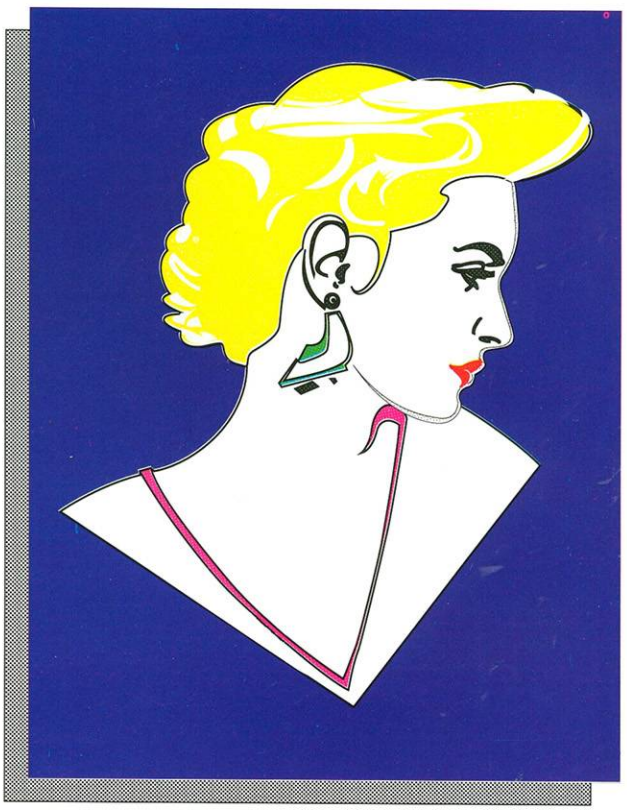
Designer-Arbeiten mit DMV-Atelier auf Ihrem PC!

DMV Atelier

Ob Firmenlogo, Illustration, Aufmacher oder Überschrift, mit **DMV-Atelier** können Sie alles in perfekter grafischer Qualität gestalten: Schriften und Bilder in beliebigen Größen drehen, spiegeln, verzerren, kopieren, skalieren und einblenden von Mustern und Farben. Ob Firmenlogo, Aufmacher oder Überschrift - alles in Top-Qualität.

Durch Fenstertechnik, Pulldown-Menüs, Symbolanzeigen und Maussteuerung ist **DMV-Atelier** einfach zu bedienen und sofort produktiv einsetzbar. Dabei sehen Sie auf dem Bildschirm genau das, was Sie auf Matrix- und Laserdrucker ausgeben oder als Datei in DTP-Programmen weiterverarbeiten können.

Mit Ihren Grafiken können Sie die bestehende Symbolbibliothek von **DMV-Atelier** erweitern und in Ihre anderen Arbeiten einfügen. In Verbindung mit den mitgelieferten Schriften stehen Ihnen außergewöhnliche Möglichkeiten zur grafischen Gestaltung zur Verfügung.



- Äußerst leistungsfähige **Design-Funktionen** mit einfacher Benutzerschnittstelle
- Umfangreiche **Grafikfunktionen**: Verschieben, Kopieren, Drehen um beliebige Winkel, Skalieren, Spiegeln, Einblenden von Mustern und Farben.
- **Schriftgestaltung** in professioneller Qualität: beliebige Größen, beliebige Anordnung und zusätzliche Grafikeffekte
- **Ausgabe** in höchster Qualität: Matrix- und Laserdrucker erzielen Effekte, die bisher nur auf Postscript-Systemen möglich waren
- **Symbolbibliothek**: Archiv mit fertigen Grafiken und Symbolen, mit Ihren eigenen Elementen erweiterbar
- **Image-Tracing-Funktion** zur Umsetzung von Vorlagen im Pixelformat (IMG-, PCX- und TIFF-Format) in Vektorgrafik
- Volle **Kompatibilität** zu den wichtigsten DTP-Programmen, wie Ventura-Publisher, Aldus-Pagemaker etc.
- Vier **Schriften** im Lieferumfang und weitere als Zusatzpaket

Systemvoraussetzungen: PC/XT/AT oder Kompatible mit 640 KByte RAM, Festplatte, Maus oder Grafiktablett als Eingabegerät empfehlenswert, vorhandener EMS-Speicher wird unterstützt. Drucker: 9-Nadel- und 24-Nadel-Drukker, HP-kompatible Laserdrucker, HP-Laserjet Plus und HP-Paintjet im Lieferumfang, weitere auf Anfrage.

* Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland 4.- DM bzw. für das Ausland 6.- DM Porto und Verpackung.
Bitte benutzen Sie die Bestellkarte.

DMV-Atelier	DM	**249,–***
Zusatzfonts 41 Schriften mit Installation	DM	**99,–***
Symbolbibliothek Zusätzliche Symbole	DM	**99,–***

Für interessante Hintergrundmotive:
Fraktal-Generator 3D DM **69,–***

DMV-Verlag • Postfach 250 • 3440 Eschwege

DMV Daten- und Medienverlag

Listings über Listings

Fluten von Einsendungen erreichen die DOS-Redaktion Tag für Tag. Ein Großteil davon ist das Ergebnis vieler Arbeitsstunden unzähliger findiger Köpfe. Manchmal bekommen wir Lösungen nie gekannter Probleme, die sich aus diversen Hardwarekonstellationen ergeben, oft aber auch schlaue Kniffe für jedermann.

Wir versuchen stets, Ihnen diese Spitzen-Programme in der DOS zugänglich zu machen, aber das gelingt nicht immer. Deshalb bekommen Sie dieses Mal eine DOS Extra voll mit Listings. Damit das Ganze nicht zu trocken wird, führen wir Sie zu Beginn des Hefts in einige Spezialthemen der Programmierung ein und zeigen Ihnen danach Tips, Tricks und Utilities, wie Sie es aus der DOS International kennen.

Erfahren Sie auf den folgenden Seiten, wie Sie mit Interrupts umgehen, residente Programme verwalten, EAN-Codes entschlüsseln und sogar den Editor von Turbo C++ programmieren. Optimieren Sie Ihre Pascal-Quelltexte in Schreibweise und Wirkung. Das Filter-Programm vereinheitlicht den Quelltext und das Prüfprogramm zeigt Ihnen, wo das Programm die meiste Zeit zubringt.

Ein paar kleine Utilities werden Sie bei der Verwaltung Ihrer Festplatte künftig nicht mehr missen wollen. Verzeichnisse und Dateien durchsuchen, löschen und erzeugen Sie mit einfachen Aufrufen.

Wenn Ihr Drucker nicht so will, wie Sie es gerne hätten, dann legen Sie einen Treiber dazwischen oder entwerfen Sie völlig neue Zeichensätze. Dabei hilft Ihnen das Siegerprogramm des Wettbewerbs »Zeichensatzeditoren«. Als zweite Anwendung finden Sie »SBackup«, ein komfortables Programm zur Datensicherung mit Menüs, Maus und allem drum und dran.

Interessieren Sie sich besonders für die grafischen Fähigkeiten Ihrer Maschine? Dann stellen Sie sich ein paar Animationen in 3D-Grafik zusammen oder bringen Sie Bézier-Kurven auf den Bildschirm. Sicher helfen Ihnen die besonders schnellen Grafikroutinen weiter, wenn Sie sie in bereits bestehende oder neue Programme einbinden.

Naturgemäß ist ein Großteil der Software in Turbo Pascal verfaßt. Häufig enthalten die Programme aber auch Assembler-Einlagen. So erfahren Sie auch etwas über die grundlegenden Techniken dieser Kombination und haben fortan keine Probleme mehr, Assembler-Anteile für Ihre eigenen Programme zu entwickeln. Ist Ihnen dann ein begnadetes Werk gelungen, dann schicken Sie es uns. So helfen Sie mit Ihrer Fachkenntnis auch anderen Lesern, die vor ähnlichen Aufgaben stehen.

Das ist auch das grundlegende Ziel der DOS. Und so hoffen wir, daß wir Ihnen auch mit diesem Heft ein gutes Stück des Wegs ebnen.



Viel Spaß beim Abtippen und Ausprobieren wünscht Ihnen



Dieter Bäumler

INHALT

DOS EXTRA
14/91

PROGRAMMIERUNG

TEML – die Makrosprache unter Turbo C++ 6

Quelltextoptimierung 10

Fehlersuche mit Registeranzeige 14

Der enträtselte Strichcode 18

Keine Angst vor großen Bäumen 24

Wer schöne Quellcodes will, muß nicht leiden 29

Solo für die »Unterbrecher« 32

Turbo auch ohne Coprozessor 45

UTILITIES

F-Tools: Kampf dem Festplattenchaos 46

Festplatte jetzt ohne Ballast 52

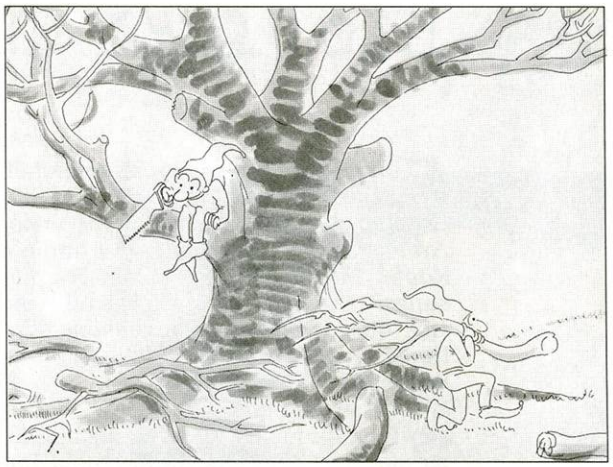
Angezapfte Hardware-Uhr 56

Umlaute für den Laser 64

Der TSR-Wächter 68

Kopieren in einem Zug 76

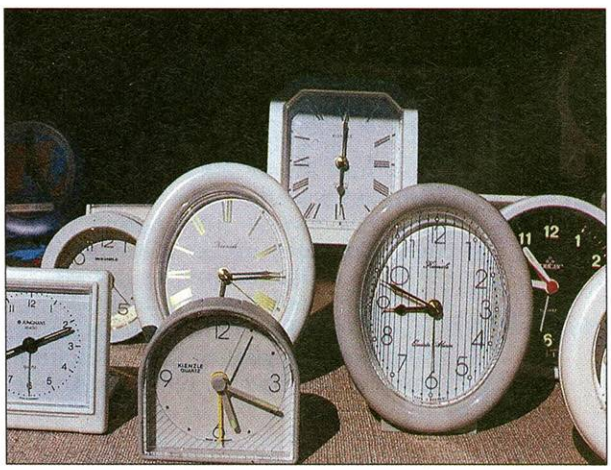
Checksummer für die Listing-Eingabe 80



Ein Teil Ihres Verzeichnisses hat für Sie keinen Nutzen mehr. Löschen Sie den kompletten Verzeichnisast mit einem Kommando.
Seite 24



Die Programmierung mit Interrupts ist nicht besonders schwer. Auch unter Pascal lösen Sie viele Probleme mit den Unterbrechern eleganter.
Seite 32



Ein kleines Programm speichert fortwährend die Zeit in einer Datei und liest sie nach dem Booten wieder
Seite 56

GRAFIK

Vom Raum auf den Bildschirm 84

Schnelle Grafiken 86

Bézier-Kurven in Turbo Pascal 90

TIPS & TRICKS

Das BIOS auf Diskette 92

Batch-Dateien werden mausfähig 93

Fremdsprachen 96

Globale MS-DOS-Operation 98

Matt dem Schnittstellenchaos 99

Wochentags-Berechnung einmal völlig anders 100

ANWENDUNGEN

Mehr Komfort mit SBackup 102

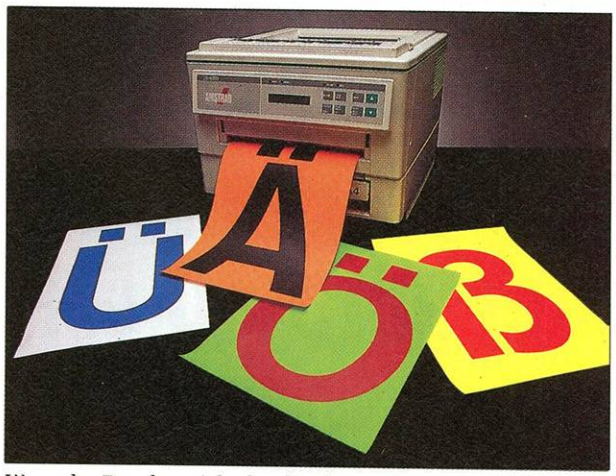
Tanz der Nadeln 139

RUBRIKEN

Editorial 3

Inhalt 4

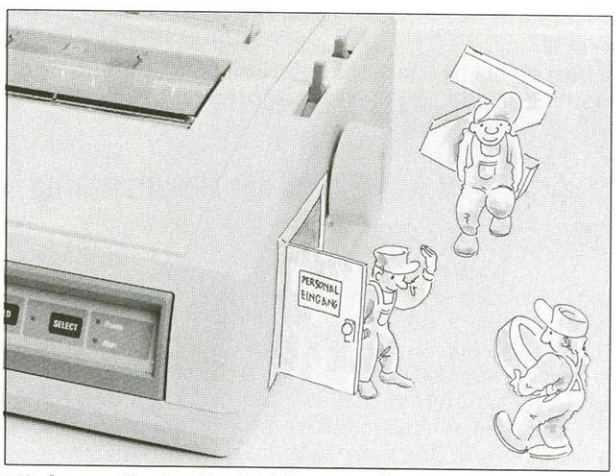
Impressum 166



Wenn der Drucker nicht das druckt, was er drucken soll, dann schalten Sie ein Programm zwischen, das Zeichencodes in andere umwandelt
Seite 64



Die meisten Backup-Programme sind überladen und wenig übersichtlich. SBackup bietet Ihnen Komfort und Geschwindigkeit bei der Datensicherung.
Seite 102



Mit dem professionellen Zeichensatzeditor generieren Sie mühelos neue Zeichensätze für Ihren 24-Nadeldrucker
Seite 139

Turbo C++

TEML – Makrosprache unter Turbo C++

Der programmierbare Editor steigert den Komfort bei der Arbeit mit Turbo C++ erheblich. Das Fehlen von Kontrollstrukturen (Schleifen und bedingte Verzweigungen) schränkt den Einsatz des Instruments zwar ein, trotzdem verfügen Sie mit »TEML« über eine leistungsfähige Makrosprache.

Kleine, schnell geschriebene Makros erleichtern die tägliche Arbeit ungemein, so daß man sich beim Entwickeln von Programmen auf das eigentliche Problem konzentrieren kann und ständig wiederkehrende Routineaufgaben dem Editor überläßt. Dieser Artikel enthält einige Tips und Tricks, die nicht im Handbuch stehen, und stellt zudem eine praktische Anwendung vor.

Im folgenden wird nun von einer integrierten Entwicklungsumgebung (IDE) ausgegangen, die auf dem Laufwerk E mit den vorgegebenen Verzeichnissen installiert wurde. Lediglich der Verzeichnisname »tc« wurde bei der Installation in »tc_plus« geändert. Pfad- und Laufwerksangaben müssen deshalb auf Ihre eigenen Verhältnisse angepaßt werden. Jeder, der mit »TEML« experimentieren möchte, sollte zuerst eine Sicherheitskopie seiner – mehr oder weniger mühsam – generierten Konfigurationsdatei anfertigen, da erzeugte Makros dort hineingeschrieben werden, beispielsweise durch:

```
copy e:\tc_plus\bin\tcconfig.tc e:\tc_plus\bin\
tcconfig.tc2
```

Sollte nun etwas passieren, kopiert man »tcconfig.tc2« einfach wieder zurück. Später wird übrigens noch eine komfortablere Lösung vorgestellt.

Die richtige Integration in die Benutzeroberfläche

Als nächstes wird der Makrocompiler in die IDE eingebaut, so daß zum Kompilieren nur die Tem-Datei im aktuellen Fenster stehen muß und alles andere automatisch abläuft. Zuerst wird unter »Options« der Menüpunkt »Full Menues« auf »On« geschaltet. Als dritter Unterpunkt ist nun »Transfer« zu erkennen. Nach Aufruf von Transfer ergibt sich eine Darstellung wie in Bild 1 gezeigt. Da der Menüpunkt »Edit« aktiviert ist, genügt es, [Enter] zu drücken, um die Transferliste mit folgendem Eintrag überarbeiten zu können:

```
Program Title: Makro Compiler
Program Path : e:\tc_plus\bin\temc
Command Line : $save cur $cap edit $edname $config
```

In Bild 2 ist der vollständige Eintrag in die Transferliste dargestellt. Zur Festlegung der Aktivierungstaste wird bei Eingabe des Programmtitels vor das »M« von »Makro Compiler« eine Tilde (»~«, [Alt-126]) gesetzt. Mit [Alt-Leertaste-M] läßt sich nun der Makrokompiler aufrufen; als Parameter werden die Namen der Dateien mit der Endung »tem« und der aktuellen Konfigurationsdatei übergeben (»$edname« und »$config«). »$config« veranlaßt die IDE außerdem, die neuen Einstellungen direkt zu übernehmen, neue Makros können also sofort ausprobiert werden. »$save cur« stellt sicher, daß immer die aktuelle Version übersetzt wird, »$cap edit« leitet die Ausgabe von »temc« in ein Fenster um. Es reicht also, im aktuellen Fenster eine Tem-Datei zu bearbeiten und den neuen Menüpunkt »Makro Compiler« zu aktivieren.

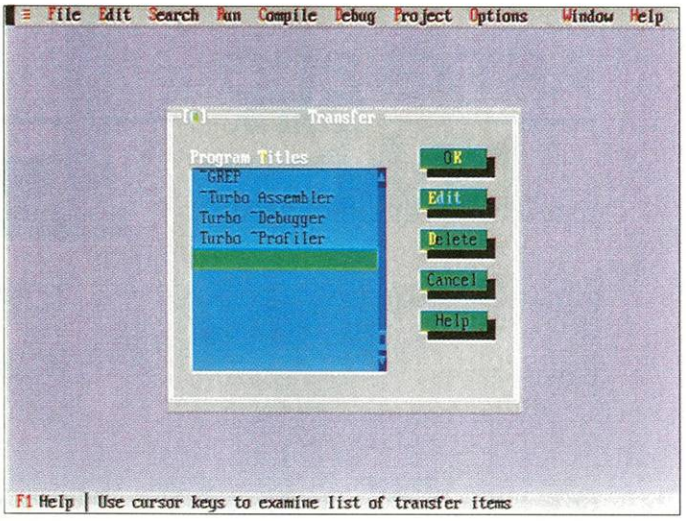


Bild 1. Nach Aufruf von »Transfer« im Menü »Options« kann ein neuer Transfereintrag festgelegt werden

Nun sollen Änderungen in der Konfigurationsdatei auch wieder rückgängig gemacht werden können, also wird die kleine Stapeldatei »nomacros.bat« (Listing 1) erzeugt und im Bin-Verzeichnis abgelegt. Der dazu gehörende Transfereintrag lautet:

```
Program Title: Makros löschen
Program Path : e:\tc\bin\nomacros
Command Line : $config
```

Bild 3 stellt die Transferliste für den Eintrag «Makros löschen« dar. Auch hier sorgt »$config« dafür, daß einerseits wirklich die aktive Konfigurationsdatei geändert wird, andererseits die Änderungen auch sofort übernommen wer-

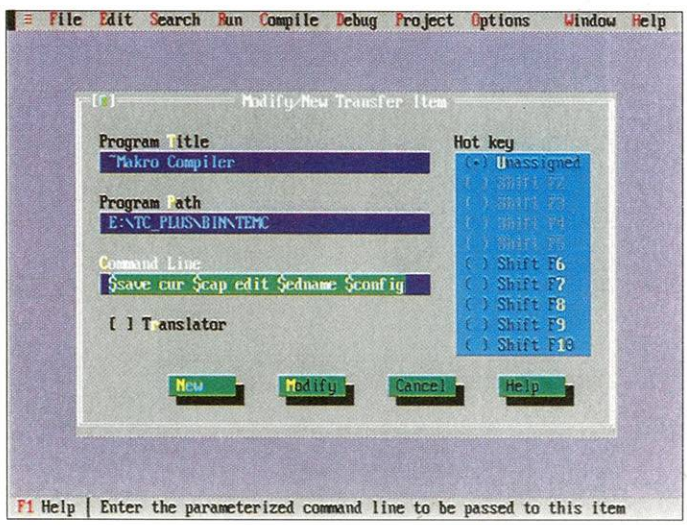


Bild 2. Ein vollständiger Transfereintrag für das neue Menü »Makro Compiler«

den. Sofern Sie, wie weiter oben angegeben, mit dem DOS-Befehl »copy« die Datei »tcconfig.tc« in »tcconfig.tc2« kopiert haben (Listing 1 baut genau auf dieser Namensgebung auf) und in der Konfigurationsdatei noch keine Makros enthalten waren, führt der Menüpunkt »Makros löschen« zur kompletten Tabula rasa; alle Makros müssen dann erst wieder neu kompiliert werden. Da die beiden neuen Menüpunkte »Makro Compiler« und »Makros löschen« nun fertiggestellt sind, kann die Makrodatei »makros.tem« (Listing 3) nun geladen und dann mit [Alt-Leertaste-M] kompiliert werden (Bild 4).

Damit ist in der Entwicklungsumgebung das Rüstzeug für die Makroprogrammierung vorhanden. Nun noch ein paar Tips für den Einstieg:

Schneller als erwartet, taucht nach einiger Zeit die Meldung »script too complex« auf. Selbst wenn die Tem-Datei erheblich gekürzt wird, kann es sein, daß diese Meldung wiederkehrt, teilweise sogar bei Scripts, die vorher einwandfrei kompiliert wurden. Deshalb sollte nach einer Kürzung vor dem erneuten Kompilieren die Funktion »Makros löschen« in der Transferliste aufgerufen werden. Wenn das Script dann immer noch »too complex« ist, ist die Fehlermeldung tatsächlich berechtigt.

Bei der Makroprogrammierung müssen sie besondere Feinheiten beachten

Wenn Cursorbewegungen andere Ergebnisse liefern als erwartet, kann das verschiedene Gründe haben. Zum einen gibt es die klaren Denk- oder Flüchtigkeitsfehler, zum Beispiel, wenn in einer Folge von zehnmal »Cursor Up« der zehnte Befehl vergessen wurde; das ist schnell einmal passiert. Dann gibt es aber auch noch die etwas gemeineren Fußangeln, ein Beispiel dafür ist folgendes:

```
LeftOfLine;
InsertText("\nHello World");
```

Was diese Befehlsfolge bewirkt, ist klar: Der Cursor wird an den linken Fensterrand gesetzt, eine Zeile eingefügt (»\n«) und dann die Zeichenfolge ausgegeben. Die Frage ist nun, wo das »Hello World« steht. Wer jetzt den linken Fensterrand vermutet, liegt nicht ganz falsch, aber je nach Situation kann der Satz auch ein ganzes Stück weiter rechts liegen. Wieso verhält sich das System anders als erwartet? Nun, einige werden es wahrscheinlich schon erraten haben, das Problem liegt in der Methode, mit der die Leerzeile eingefügt wurde. Das »\n« wird als Drücken der Taste »Enter« interpretiert, das heißt, ist im Editor der Modus für automatische Einrückung eingeschaltet, hält sich auch TEML daran und rückt den Cursor nach rechts.

Vorsicht ist geboten, wenn die aktuelle Cursorposition gespeichert, aber kein Marker verwendet werden soll. Man benutzt also »SetPrevPos« und springt später mit »MoveToPrevPos« zurück. Im Handbuch ist beschrieben, daß einige Makros »PrevPos« verändern, aber man hat Vorsicht walten lassen, und alles funktioniert prima. Trotzdem sollte in der Dokumentation die Benutzung von »PrevPos« besonders erwähnt werden, weil es sonst leicht passieren kann, daß einige Tage später bei einer Änderung eben diese Tatsache übersehen wird. Je nach Größe des Makros kann der zeitliche Aufwand für die dann fällige Fehlersuche weitaus größer sein als der für eine simple Kommentarzeile. Und wer ganz sicher gehen will, benutzt zum Sichern der Cursorposition einfach die wenig gebrauchten Marker 8, 9 und 0.

Wenn eine Fehlfunktion nicht durch Denkfehler, Seiteneffekte anderer Makros oder bestimmter Einstellungen zu erklären ist, kann es sein, daß der Fehler in TEML steckt. Listing 2 (»test.tem«) zeigt ein Problem, das sich dem Autor stellte.

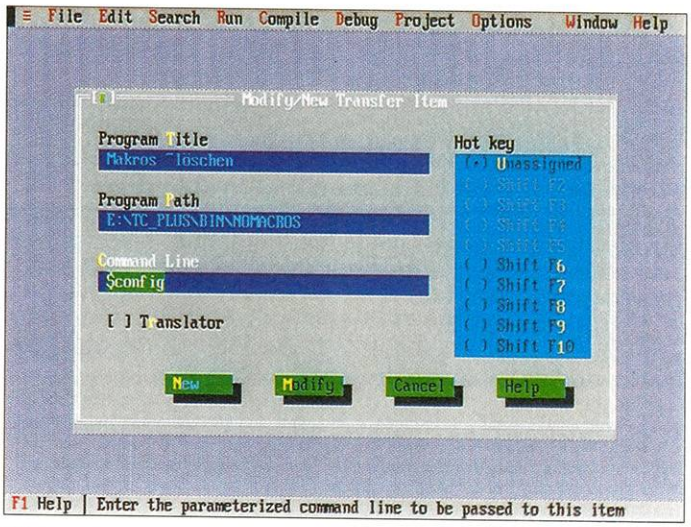


Bild 3. Ein weiteres Menü, »Makros löschen«, wird erzeugt

Nachdem das Script kompiliert ist, öffnet man mit [F10-F-N] ein neues Fenster. Beim Drücken von [Alt-K] erlebt man eine Überraschung: Der Cursor steht am Zeilenende und nicht, wie vorgesehen, eine Position weiter rechts. Das Makro, das auf [Alt-L] liegt, bringt das gleiche Ergebnis wie die erste Version: TEML ignoriert also einfach die Cursor-Right-Anweisung. In diesem Fall läßt sich das Problem umgehen, indem man den Befehl einfach nach dem Sprung zur Marke 0 ausführt (siehe Makro [Alt-M] in Listing 2), und es ist anzunehmen, daß es bei anderen Problemen ähnlich einfache Lösungen gibt.
Man sollte allerdings mit der Hypothese »... und schon wieder ein Fehler in TEML« vorsichtig sein, wie das Hello-World-Beispiel beweist: Die Hauptfehlerquelle ist nach wie vor der Anwender.

Ein praxisorientiertes Makro

Nun eine Anwendung aus dem Bereich der Programmentwicklung. Eigentlich sollten Programme und Funktionen schon beim Schreiben dokumentiert werden. Das sieht jeder ein, doch kaum jemand macht sich tatsächlich die Mühe, es konsequent durchzuhalten. Mit den hier vorgestellten Makros bleibt es jedoch nicht nur bei guten Vorsätzen. Listing 3 (»macros.tem«) stellt folgende fünf Tastaturmakros zur Verfügung:

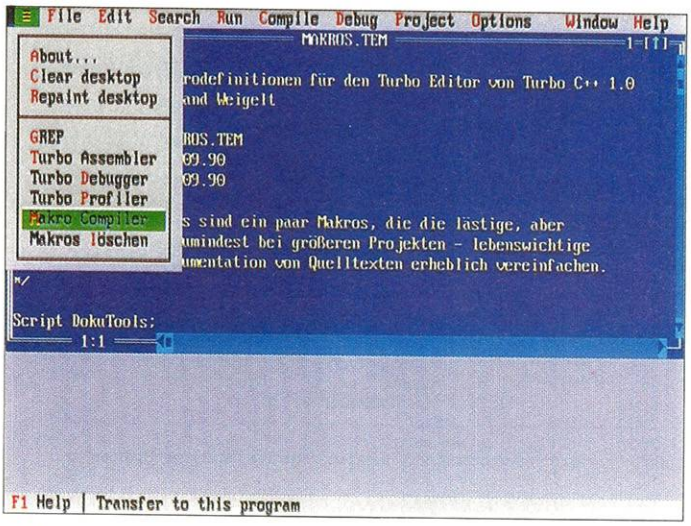


Bild 4. Die neuen Menüpunkte warten auf ihren Einsatz bei »makros.tem«

- [Alt-K] erzeugt einen Programmkopf, in den nur noch ein paar Informationen eingetragen werden müssen. Der Titel in Listing 3 wurde übrigens damit erzeugt.
- [Alt-L] zieht eine Linie, die in eigenen Kommentaren verwendet werden kann.
- [Alt-T] fügt eine Kommentarzeile ein, die der optischen Trennung von zwei Sinnabschnitten dient.
- [Alt-I] übernimmt die Arbeit, die Include-Anweisungen für die Header-Dateien bei jedem Programm neu einzugeben. Hier muß das Script natürlich an die eigenen Bedürfnisse angepaßt werden.
- [Alt-B] erzeugt die Dokumentation für eine Funktion. Einfach den Cursor auf die Zeile der Funktionsdeklaration fahren und das Makro aufrufen. Ein Beispiel:

```
int *Test_Funktion(int test1,char *test2)
{
...
}
```

Daraus entwickelt sich eine Zeichenfolge, wie Sie im Kasten sehen.

Das letzte Makro zeigt recht deutlich, was bei geschickter Programmierung aus TEML herausgeholt werden kann und wo die Grenzen liegen. Es ist logisch, daß das Makro bei einer Deklaration wie zum Beispiel

```
unsigned char Test2(<Parameter>)
```

kapitulieren muß. Da nicht abgefragt werden kann, auf welchem Buchstaben oder Wort der Cursor steht, wird hier das »char« als Funktionsname interpretiert.

Trotzdem befindet sich besonders dieses Makro beim Autor im täglichen Einsatz. Es ist ohnehin empfehlenswert, in C-Quelltexten komplizierte Variablentypen mit Hilfe von »#define« zu vereinfachen, zum Beispiel

```
#define Byte unsigned char
```

```
/*------------------------------
** Name : Test_Funktion
** Aufgabe :
**
** Parameter: int test1,char *test2
** [Enter] : int *
**
** Bemerkung:
*/

int *Test_Funktion(int test1,char *test2)
{
...
}
```

Das spart Tipparbeit, ist besser lesbar und kostet in der späteren Exe-Datei kein Byte mehr.

Versuchen Sie, beim Erstellen eigener Makros entweder ständig wiederkehrende Arbeitsabläufe herauszufinden oder komplexere Befehlsfolgen zusammenzufassen. Die so entstandenen Makros sollten, unter anderem aus den schon beschriebenen Gründen, sorgfältig in verschiedenen Betriebszuständen (zum Beispiel bezüglich des Insert-Modus) getestet werden. Außer der Entlastung von Routinetätigkeiten und der Zeitersparnis haben Makros dann noch einen weiteren Vorteil: Sie verhindern Fehler, die sich bei manueller Eingabe hätten einschleichen können.

Die Makrosprache TEML ist ohne die anfangs erwähnten Kontrollstrukturen in bestimmten Situationen oft überfordert. Vielleicht bringt die nächste Compilerversion hier etwas Neues; in der Zwischenzeit gibt es jedoch immer noch genügend Gelegenheiten, dieses praktische Hilfsmittel sinnvoll einzusetzen. (Roland Weigelt/bj)

```
[FD1]    1: @echo off
[7NØ]    2: rem
[T48]    3: rem  Diese Datei löscht alle Makrodefinitio
            nen
[7NØ]    4: rem
[4H1]    5: copy %12 %1
```

Listing 1. Die Stapeldatei »nomacros.bat« kopiert den Inhalt von »tcconfig.tc2« zurück in »tcconfig.tc« und löscht damit die noch nicht in »tcconfig.tc2« aufgenommenen Makros

```
[Y32]    1: Script Test;
[ØØØ]    2:
[U32]    3: /* Version 1 */
[MY1]    4: Alt-K :Begin
[4X4]    5:          RightOfLine;      /* Zeilenende */
[436]    6:          CursorRight;      /* eins nach rech
            ts */
[X66]    7:          SetMark(Ø);       /* Position speic
            hern */
[YX6]    8:          HomeCursor;       /* irgendwohin sp
            ringen */
[HG3]    9:                            /* (hier Textanfa
            ng) */
[A67]    1Ø:         MoveToMark(Ø);    /* Und zur alten
            Position */
[ZNØ]    11:       End;
[ØØØ]    12:
[5Ø7]    13: /* Version 2 macht das Gleiche wie Version
            2 */
[PY1]    14: Alt-L :Begin
[752]    15:          RightOfLine;
[SL4]    16:                            /* CursorRight we
            ggelassen */
[5X1]    17:          SetMark(Ø);
[P42]    18:          HomeCursor;
[6C2]    19:          MoveToMark(Ø);
[ZNØ]    2Ø:        End;
[ØØØ]    21:
[LØ7]    22: /* Version 3 liefert das richtige Ergebnis
            */
[SY1]    23: Alt-M :Begin
[752]    24:          RightOfLine;
[5X1]    25:          SetMark(Ø);
[P42]    26:          HomeCursor;
[6C2]    27:          MoveToMark(Ø);
[P92]    28:          CursorRight;
[ZNØ]    29:        End;
```

Listing 2. Die Makrodatei »test.tem« weist auf einen Fehler in »TEML« hin

```
[W6Ø]    1: /*
[A5B]    2: ** Titel       : Makrodefinitionen für den
            Turbo Editor von Turbo C++ 1.Ø
[1L4]    3: ** Autor       : Roland Weigelt
[F64]    4: ** Dateiname   : MAKROS.TEM
[W7A]    5: ** Bemerkung   : Dies sind ein paar Makros,
             die die lästige, aber
[AAA]    6: **               - zumindest bei größeren P
            rojekten - lebenswichtige
[6EA]    7: **               Dokumentation von Quelltex
            ten erheblich vereinfachen.
[MX4]    8: ** (c)1991 DMV Widuch GmbH & Co. KG
[17Ø]    9: */
[ØØØ]    1Ø:
[BØ3]    11: Script DokuTools;
[ØØØ]    12:
[4G2]    13: /*---- Makros ----*/
[ØØØ]    14:
[XX1]    15: Macro Linie
[E42]    16:   InsertText(
[D98]    17:   "-----------------------------------
            ---------------------------------");
[ZNØ]    18: End;
[ØØØ]    19:
[VT2]    2Ø: Macro Trennzeile
[LK2]    21:   SetInsertMode;
[LØ2]    22:   LeftOfLine;
[C93]    23:   InsertText("/**/\n");
[EP1]    24:   CursorUp;
[P92]    25:   CursorRight;
[P92]    26:   CursorRight;
[F11]    27:   Linie;
[LØ2]    28:   LeftOfLine;
[ZNØ]    29: End;
[ØØØ]    3Ø:
[N53]    31: Macro Programmkopf
[KM2]    32:   HomeCursorRaw;
[LK2]    33:   SetInsertMode;
[1A5]    34:   InsertText("/*\n** Titel       :\n");
[EP1]    35:   CursorUp;
```

```
[752]    36:    RightOfLine;
[5X1]    37:    SetMark(Ø);
[R42]    38:    CursorDown;
[LØ2]    39:    LeftOfLine;
[2NA]    4Ø:    InsertText("** Autor       :\n**\n** File
           name    :\n** begonnen am :\n");
[ZL9]    41:    InsertText("** Version vom :\n**\n** Beme
           rkung   :\n*/\n\n");
[6C2]    42:    MoveToMark(Ø);
[P92]    43:    CursorRight;
[ZNØ]    44:  End;
[ØØØ]    45:
[645]    46:  Macro Bemerkung_zur_Funktion
[LK2]    47:    SetInsertMode;
[LØ2]    48:    LeftOfLine;
[3V1]    49:    WordRight;
[5X1]    5Ø:    SetMark(Ø);
[D42]    51:    SetBlockBeg;
[3V1]    52:    WordRight;
[RX1]    53:    SetMark(9);
[X42]    54:    CursorLeft;
[252]    55:    SetBlockEnd;
[LØ2]    56:    LeftOfLine;
[TV2]    57:    InsertText("\n");
[EP1]    58:    CursorUp;
[LØ2]    59:    LeftOfLine;
[N73]    6Ø:    InsertText("/*\n");
[EP1]    61:    CursorUp;
[752]    62:    RightOfLine;
[F11]    63:    Linie;
[R42]    64:    CursorDown;
[LØ2]    65:    LeftOfLine;
[C57]    66:    InsertText("** Name      :\n** Aufgabe   :\
           n**\n");
[1RA]    67:    InsertText("** Parameter:\n** Return    :\
           n**\n** Bemerkung:\n*/\n");
[EP1]    68:    CursorUp;
[EP1]    69:    CursorUp;
[EP1]    7Ø:    CursorUp;
[EP1]    71:    CursorUp;
[EP1]    72:    CursorUp;
[EP1]    73:    CursorUp;
[EP1]    74:    CursorUp;
[EP1]    75:    CursorUp;
[752]    76:    RightOfLine;
[P92]    77:    CursorRight;
[UT1]    78:    CopyBlock;
[R42]    79:    CursorDown;
[R42]    8Ø:    CursorDown;
[R42]    81:    CursorDown;
[752]    82:    RightOfLine;
[NX1]    83:    SetMark(8);
[FC2]    84:    MoveToMark(9);
[D42]    85:    SetBlockBeg;
[752]    86:    RightOfLine;
[X42]    87:    CursorLeft;
[252]    88:    SetBlockEnd;
[EC2]    89:    MoveToMark(8);
[P92]    9Ø:    CursorRight;
[UT1]    91:    CopyBlock;
[R42]    92:    CursorDown;
[752]    93:    RightOfLine;
[NX1]    94:    SetMark(8);
[6C2]    95:    MoveToMark(Ø);
[252]    96:    SetBlockEnd;
[LØ2]    97:    LeftOfLine;
[D42]    98:    SetBlockBeg;
[EC2]    99:    MoveToMark(8);
[P92]   1ØØ:    CursorRight;
[UT1]   1Ø1:    CopyBlock;
[252]   1Ø2:    SetBlockEnd;
[EP1]   1Ø3:    CursorUp;
[EP1]   1Ø4:    CursorUp;
[EP1]   1Ø5:    CursorUp;
[ZNØ]   1Ø6:  End;
[ØØØ]   1Ø7:
[W33]   1Ø8:  Macro IncludeFiles
[LØ2]   1Ø9:    LeftOfLine;
[UBC]   11Ø:    InsertText("#include <stdio.h>\n#include
           <stdlib.h>\n"); /* ggfs ändern */
[CA9]   111:    InsertText("#include <string.h>\n#include
            <conio.h>\n");
[BG8]   112:    InsertText("#include <dos.h>\n#include <m
           em.h>\n");
[ZNØ]   113:  End;
[ØØØ]   114:
[9P4]   115:  /*---- Tastendefinitionen ----*/
[ØØØ]   116:
[TE3]   117:  Alt-K :ProgrammKopf;
[PY1]   118:  Alt-L :Begin
[LK2]   119:           SetInsertMode;
[F11]   12Ø:           Linie;
[ZNØ]   121:         End;
[E23]   122:  Alt-T :TrennZeile;
[KE5]   123:  Alt-B :Bemerkung_zur_Funktion;
[PC3]   124:  Alt-I :IncludeFiles;
```

Listing 3. Die Makrodatei »macros.tem« enthält fünf einsatzfähige Makros, die lediglich den eigenen Systemverhältnissen angepaßt werden müssen

BATH BEACH
SOFTWARE-SERVICE

PUBLIC DOMAIN 150

Translator übersetzt englische Programm-Dokumentationen in Deutsch, **Lightning-Press** DTP-Desktop-Publishing, Fantastic **As Easy As** sehr gutes Tabellenkalkulationsprogramm, Bestseller **Prophet-Lotto** Statistikprogramm für Lotterieziehungen, **3D-Chess** dreidimensionales Schachprogramm, **PC-Write** sehr gute Textverarbeitung mit vielen Möglichkeiten, **Free-File** sehr gute Datenverwaltung mit vielen Möglichkeiten, **Composer** Musik komponieren, Noten schreiben und ausspielen, **PC-Akt** erzeugt digitalisierte Aktbilder auf ihrem PC, **Dancad-3D** dreidimensionales CAD-Zeichenprogramm mit Animation, **PC-KWIK** beschleunigt ihr Diskettenlaufwerk, **Mahjongg** sehr gutes chinesisches Spiel mit guter Grafik **Megagopoly** Spiel, wie erreiche ich eine Million, **Nuke New York** zeigt einen nuklearen Angriff und die Auswirkungen auf New York **Designer** ein Zeichenprogramm, das bewegliche Bilder erstellt **Black Jack** spielen wie im Casino **Horoskop** erstellt Ihr persönliches Horoskop **Kniffel** Kartenspiel **Mastermind** populäres Strategiespiel **Star-Traders** Händlerspiel, Aktien kaufen im Weltraum **Etikett** erstellt Etiketten in verschiedenen Formaten und druckt sie aus **Flugsimulator** simuliert Jet-Flüge **PC-Calc** populäre Tabellenkalkulation **Copy-All** nun können Sie fast alle geschützten Programme kopieren **PC-Window** PC-Fenstertechnik **Kalender** erstellt einen Kalender mit Bildschirm/Druck **Printum** schaltet den Drucker in andere Schriftarten um **Bastools** Basic-Tools **Your Menu** erstellt Ihr eigenes Menü zum Selbst-Ausführen **Form-Master** umfangreicher Formular-Generator **DOS-Help** zeigt und erklärt alle DOS-Befehle**StockCharting** Aktienverwaltung,Kauf-/Verkaufssignal(graf. Darstellung) **Biorythm** Biorythmen **Finance** Finanzen und Berechnungen**Taskplan** Zeitplanungsprogramm **Cover** druckt Diskettenhüllen mit Inhaltsverzeichnis **VCR-D-Base** verwaltet Video-Casetten und druckt Aufkleber **Bookinventory** Inventurprogramm für bis zu 2000 Bücher

150 Programme für nur 130.- DM!

P C -Lock schützt ihre Festplatte mit Passwort **Best Utilities** jede Menge Hilfsprogramme **Cataloging** erstellt einen Diskettenkatalog **FB-Copy** ein sehr schnelles und komfortabeles Kopierprogramm **Virus Check** Computerviren-Untersuchungsprogramm **Backgammon** Backgammonspiel **Space-Invaders** sehr schönes, bekanntes Arcade-Spiel **PC-Foto** Fotoverwaltungsprogramm incl. Druck von Dia-Aufklebern **Packdisk** macht mehr Platz auf Ihrer Festplatte **Compress** Datenkompakter, der Platz auf der Diskette spart **Chasm** Super Assembler-Programmiersprache **plus weitere 75 Public Domain- und Shareware-Programme.** *Hilfs-, Anwendungs- und Spielprogramme (BAS-, COM- und EXE-Startdateien) für IBM PC/XT/AT mit VGA, EGA-, CGA- und Herculeskarte.Alle Programme wurden auf Viren geprüft.*

Schnellversand am Tag des Bestelleingangs. Großer INFO-Katalog mit Programm-Beschreibungen und zusätzlichen Angeboten liegt jeder Bestellung bei.

Inklusive deutscher Anleitung zum Starten aller Programme!



© UESSFELD & Partner LANDAU

Bath Beach

Software-Service GmbH

Speckstraße 23, D-6749 Schweighofen

BTX: * BATH # • FAX:(06342)6216 • Tel: (06343) 5818

BESTELL-COUPON

Ja, bitte senden Sie mir umgehend das PC-Paket "Public Domain 150" zum Discount-Preis von 130.-DM. **Bei Vorkasse keine Versandkosten.**
Bei Nachnahme: DM 10.- Versandkosten. Den Gesamtbetrag (Endpreis) bezahle ich wie angekreuzt. (Ausland nur Vorkasse: Scheck oder Bar).
Zusätzlich erhalte ich **kostenlos** den ausführlichen INFO-Katalog.

Endpreis: ☐ 5 1/4"Disketten ☐ Scheck/Bar DM 130.- ☐ Per Nachnahme DM 140.-
Endpreis: ☐ 3,5"Disketten ☐ Scheck/Bar DM 150.- ☐ Per Nachnahme DM 160.-
(3,5"Zoll-Disketten Material-Mehrpreis : 20.-)

Name:
Straße: Computer-Typ:
PLZ/ORT: Grafik-Karte:
Telefon:
Datum /Unterschrift
(Bei Minderjährigen der Erziehungsberechtigte) Geburtsdatum DOEX 14

Turbo Pascal ab Version 5.0

Quelltext-optimierung

Bei größeren Programmen kommt es häufig vor, daß man gerne wissen möchte, welche Prozeduren in welcher Reihenfolge wie oft aufgerufen werden. Grund dafür ist, diejenigen Prozeduren zu ermitteln und dann zu optimieren, die am häufigsten zur Ausführung gelangen. Daher wurde die Unit »stat« für Turbo Pascal entwickelt.

Bindet man die Unit mit »uses stat« in eigene Programme ein, wird zur Laufzeit ein Protokoll darüber geführt, welche Prozeduren (und Funktionen) aufgerufen werden. Später kann dann ein Analyseprogramm eine Auswertung bezüglich der Häufigkeit der einzelnen Prozeduren durchführen. Das Utility besteht deshalb aus zwei Komponenten:

- der Unit »stat.pas« (Listing 1), die in das zu analysierende Programm eingebunden wird, und
- dem Auswertungsprogramm »ana.pas« (Listing 2).

Es wäre zwar ohne weiteres denkbar, die Auswertung in die Unit »stat« mit aufzunehmen, dies würde dann aber das zu analysierende Programm unnötig belasten (Datensegment, Heap). Daher ist der Zweiteilung der Vorzug zu geben.

Beim Compiler Turbo Pascal kann der Programmierer durch Setzen des Compilerschalters »Stack checking« auf »On« den Compiler veranlassen, bei jedem Prozedureingang – dieser wird auch Prolog genannt – den Aufruf einer Systemroutine zu codieren. Diese Systemroutine überprüft dann zur Laufzeit des Programms, ob genug Platz auf dem Stapel vorhanden ist, um alle lokalen Variablen der eben aufgerufenen Prozedur aufzunehmen. Üblicherweise wird dieser Compilerschalter nur während der Entwicklung gesetzt. Hat sich das Programm als (vorläufig) fehlerfrei gezeigt, wird auf Stapelprüfung verzichtet. An dieser Routine zur Stapelprüfung setzt nun die Unit »stat.pas« (Listing 1) an. Sie ermittelt zunächst die Adresse der angesprochenen Systemroutine. Dies macht sie, indem sie bei einer nicht weiter benutzten Prozedur »dummy« im Code nachsieht, welche Adresse im Prolog aufgerufen wird. Diese Adresse besteht aus einem Segment und einem Offset, die in den Variablen »chk_seg« und »chk_off« gespeichert werden. Danach überschreibt die Unit in der Systemroutine die ersten Befehle mit einem Sprungbefehl zu der Prozedur »my«, die der Unit »stat« angehört. Jedesmal, wenn beim Programmablauf eine Prozedur oder Funktion aufgerufen wird, bei der zur Compilationszeit die Stapelprüfung eingeschaltet war, wird nun die Systemroutine aufgerufen und von dort zur Prozedur »my« gesprungen. Diese prüft nun aber nicht den Stapel, sondern schreibt die Adresse der zuvor aufgerufenen Prozedur in eine Protokolldatei mit dem Namen »proto«. Die Adresse dieser Prozedur steht, um 10 oder 11 Byte erhöht, auf dem Stapel als Rückkehradresse. Vor dem Aufruf der Systemroutine zur Überprüfung des Stapels sind im Prozedurprolog Befehle codiert, welche, den Aufruf mitgerechnet, entweder 10 oder 11 Byte belegen (Bild).

Ferner legt die Unit »stat« bei der Initialisierung eine kleine Textdatei mit dem Namen »proto.inf« an. In dieser Datei steht neben dem Namen des zu analysierenden Programms auch die Segmentadresse der Unit »stat«.

Beim Verlassen des zu analysierenden Programms wird die Exit-Prozedur durchlaufen. Dazu gibt es bei Turbo Pascal die Systemvariable »ExitProc«, welche auf die Exit-Prozedur zeigt. In der Unit »stat.pas« steht die Prozedur »my_exit« in dieser Kette der Exit-Prozeduren. Bei einem Programmabbruch wird dann die Prozedur »my_exit« und anschließend die übliche Exit-Prozedur aufgerufen. Das Handbuch zu Turbo Pascal erläutert dies näher. Die Exit-Prozedur zu dieser Unit macht nichts anderes, als die Protokolldatei zu schließen.

Das Analyseprogramm erhält als Eingabedateien:

- die Protokoll-Informationsdatei »proto.inf«; sie wird beim Lauf des zu analysierenden Programms erzeugt.
- die eigentliche Protokolldatei »proto«; sie wird ebenfalls beim Lauf des zu analysierenden Programms generiert.
- die zum Programm gehörende Map-Datei. Vergessen Sie auf keinen Fall, diese zu erzeugen. Dazu muß vor dem Kompilieren die Option »Debug Info« auf »On« und die Linker-Option »Mapfile« auf »Detailed« gestellt werden.

Das Analyseprogramm arbeitet die Protokolldatei ab und sucht für jede angegebene Adresse heraus, welcher Prozedur sie entspricht. Bei globalen Prozeduren, die entweder im Hauptprogramm stehen oder im Interface-Teil ihrer Unit deklariert sind, ist dies leicht anhand der Publics-Liste zu handhaben. Bei lokalen Prozeduren, das sind Prozeduren, die innerhalb einer anderen Prozedur deklariert wurden oder nur innerhalb einer Unit gültig sind, muß die Liste der »Line Numbers« durchsucht werden. In beiden Fällen ist

```
var a : byte;

procedure keine_Variablen;
begin
writeln;
end;

procedure mit_Variablen;
var x : word;
begin
writeln;
end;
```

Der Code zu diesen Prozeduren sieht so aus:

```
1DAB:0000 55            PUSH  BP          ;
1DAB:0001 89E5          MOV   BP,SP       ; keine lokalen
1DAB:0003 31C0          XOR   AX,AX       ; Variablen
1DAB:0005 9A4402B01D    CALL  1DB0:0244   ; Stack-Check
1DAB:000A C6063C0000    MOV   BYTE PTR
                              [003C],00
1DAB:000F 89EC          MOV   SP,BP
1DAB:0011 5D            POP   BP
1DAB:0012 C3            RET               ; Ende von Proz.1
1DAB:0013 55            PUSH  BP
1DAB:0014 89E5          MOV   BP,SP       ; zwei Byte für
1DAB:0016 B80200        MOV   AX,0002     ; lokale Variable
1DAB:0019 9A4402B01D    CALL  1DB0:0244   ; Stack-Check
1DAB:001E 83EC02        SUB   SP,+02
1DAB:0021 C6063C0001    MOV   BYTE PTR
                              [003C],01
1DAB:0026 89EC          MOV   SP,BP
1DAB:0028 5D            POP   BP
1DAB:0029 C3            RET               ; Ende von Proz.2
```

Im Falle der ersten Prozedur würde »$1dab:$000a«, bei der zweiten »$1dab:$001e« auf dem Stapel als Rückkehradresse stehen. Im ersten Fall müßte 10, im zweiten 11 subtrahiert werden, um an die Adresse des Prozeduranfangs zu kommen.

Der Aufbau des Prozedur-Prologs

noch die Hürde zu nehmen, daß die Segmentadresse zur Laufzeit von jener, welche in der Map-Datei steht, verschieden ist. Deshalb wird die Segmentadresse der Unit »stat« in der Map-Datei ermittelt und von dieser zur Laufzeit subtrahiert. Aus diesem Grunde wurde sie in die Informationsdatei eingetragen. Die Differenz, in der Variablen »rel« gespeichert, ist die Verschiebung der Units zur Laufzeit gegenüber den Werten in der Map-Datei. Sie wird von allen aus dem Protokoll eingelesenen Adressen subtrahiert, um die gleichen Segmentadressen wie in der Map-Datei zu erhalten. Während der Analyse wird eine Tabelle für alle aufgerufenen Prozeduren geführt und die Häufigkeit der Aufrufe gezählt. Abschließend erfolgt das Sortieren nach der Häufigkeit und die Ausgabe in Form einer Tabelle.

»demo.pas« verdeutlicht die Arbeitsweise der Units

Am besten lernen Sie die Funktionsweise durch das Demoprogramm »demo.pas« (Listing 3) kennen. Beachten Sie dabei folgende Punkte:
- »Stack Checking« muß eingeschaltet sein,
- ebenso »Debug Info« und
- »Map File« muß auf »Detailed« gesetzt sein.

Punkt 1 ist notwendig, um zur Laufzeit überhaupt die Prozedur »my« zu erreichen. Die Punkte 2 und 3 sind sinnvoll, um herausfinden zu können, welche Prozedur sich hinter der Adresse versteckt.
Startet man das Programm »demo.exe«, werden zwei Dateien mit den Namen »proto.inf« und »proto« erzeugt. Die Datei »proto« sieht so aus:

```
Programmname:
D:\ST\DEMO.EXE
Unit STAT:
4650
```

In der zweiten Zeile stehen der Pfad und der Name des Programms, das untersucht wird. Diese Information ist für das Analyseprogramm notwendig, um die zugehörige Map-Datei bestimmen zu können. In der vierten Zeile steht die Segmentadresse der Unit »stat«. Diese Information wird vom Analyseprogramm verwendet, um die Segmente der übrigen Units bestimmen zu können.
Die eigentliche Protokolldatei »proto« enthält die Segmente und Offsets der zur Laufzeit aufgerufenen Prozeduren, und zwar in der Reihenfolge der Aufrufe. Damit diese Datei nicht zu lang wird, wurde sie statt als Textdatei als Binärdatei angelegt. Schaut man sie sich mit dem Debugger an, kann man erkennen, daß zu jeder Adresse 4 Byte gehören: 2 Byte für den Offset und 2 Byte für das Segment der jeweiligen Prozedur.
Das in diesem Artikel beschriebene Programm »ana« liefert folgende Ausgabe:

```
Laufzeitprofil zu DEMO.EXE
0000:0091 3 ( EINGABE )
0000:005D 2 ( AUSGABE )
0000:0005 1 ( Zeile 12 in Unit DEMO.PAS )
0000:0036 1 ( Zeile 17 in Unit DEMO.PAS )
```

Die Prozedur »EINGABE« wurde dreimal aufgerufen, »AUSGABE« zweimal etc. Lokale Prozeduren stehen nicht als Publics in der Map-Datei, daher kann – anhand dieser – nur die Zeilennummer und die zugehörige Unit bestimmt werden. Es wäre jedoch mit einem gewissen Aufwand realisierbar, die zugehörigen Quelltexte zu durchsuchen.
Eine weitere Verbesserung wäre bezüglich der Zeitmessung denkbar. Dazu müßte das Prozedurende, auch Epilog genannt, mit einem Prozeduraufruf belegt werden, ähnlich wie dies im Prolog geschieht. Da beim Verlassen einer Prozedur oder Funktion jedoch keine Kontrollroutine codiert wird, ist dies etwas schwieriger. Eventuell könnte nach der typischen Byte-Folge des Epilogs gesucht werden. Dann müßte entweder beim Prozedureintritt und -verlassen jeweils die Systemzeit protokolliert oder, was etwas schneller wäre, die Differenz bereits zur Laufzeit berechnet und protokolliert werden. Das Berechnen der Differenz hätte lediglich den Nachteil, daß pro Aufrufstufe einige Bytes zusätzlich auf dem Stapel benötigt würden, um die Uhrzeit des Prozedureintritts in einer lokalen Variablen zu speichern.

(Peter Tiemann/hs)

```
[GE3]    1: { Programm stat.pas
[KD4]    2:   Funktion: Unit für ana.pas
[EK3]    3:   Sprache: Turbo Pascal
[PD3]    4:   Autor: Peter Tiemann
[P35]    5:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    6:
[FT1]    7: UNIT stat;
[ØØØ]    8:
[C91]    9: {$S+,F+}
[ØØØ]   1Ø:
[SK1]   11: INTERFACE
[ØØØ]   12:
[4L2]   13: IMPLEMENTATION
[ØØØ]   14:
[A64]   15: VAR trace    : FILE OF WORD;
[A93]   16:   old_exit : POINTER;
[ØØØ]   17:
[NS6]   18: { Die Prozedur DUMMY wird nie aufgerufen,
[L37]   19:   sondern nur benutzt, um die Adresse der
[K16]   2Ø:   Stack-Check-Routine zu ermitteln.}
[ØW2]   21: PROCEDURE dummy;
[ØØØ]   22:
[MUØ]   23: BEGIN
[R21]   24:   Write;
[ZNØ]   25: END;
[ØØØ]   26:
[NYØ]   27: {$S-}
[ØØØ]   28:
[Y72]   29: PROCEDURE my;
[ØØØ]   3Ø:
[KM2]   31: VAR   s,o : WORD;
[ØØØ]   32:
[MUØ]   33: BEGIN
[AB4]   34:   o:= MemW [SSeg:SPtr+8]-1Ø;
[P44]   35:   s:= MemW [SSeg:SPtr+1Ø];
[PE3]   36:   IF Mem[s:o] = $89 THEN
[G41]   37:     Dec(o);
[T23]   38:   Write(trace, s, o);
[ZNØ]   39: END;
[ØØØ]   4Ø:
```

```
[R83]   41: PROCEDURE my_exit;
[ØØØ]   42:
[MUØ]   43: BEGIN
[V42]   44:   Close(trace);
[MK3]   45:   ExitProc:= old_exit;
[ZNØ]   46: END;
[ØØØ]   47:
[LYØ]   48: {$S+}
[ØØØ]   49:
[EP3]   5Ø: PROCEDURE init_unit;
[ØØØ]   51:
[8R6]   52: VAR chk_ofs,        { Offset und Segment }
[CM6]   53:     chk_seg : WORD; { der Systemroutine  }
[MV5]   54:     f : TEXT;       { für das Protokoll  }
[ØØØ]   55:
[MUØ]   56: BEGIN
[CN3]   57:   Assign(f, 'proto.inf');
[YY1]   58:   Rewrite(f);
[WH4]   59:   WriteLn(f,'Programmname:');
[GW3]   6Ø:   WriteLn(f,ParamStr(Ø));
[MW3]   61:   WriteLn(f,'Unit STAT:');
[RM2]   62:   WriteLn(f,CSeg);
[KD1]   63:   Close(f);
[XN3]   64:   Assign(trace, 'proto');
[1P2]   65:   Rewrite(trace);
[EF3]   66:   old_exit:= ExitProc;
[6M3]   67:   ExitProc:= @my_exit;
[PP6]   68:   chk_ofs:= MemW [Seg(dummy):Ofs(dummy)+6];
[LN6]   69:   chk_seg:= MemW [Seg(dummy):Ofs(dummy)+8];
[XU6]   7Ø:   Mem [chk_seg:chk_ofs]:= $EA;   {jump far}
[FU5]   71:   MemW[chk_seg:chk_ofs+1]:= Ofs(my);
[CU5]   72:   MemW[chk_seg:chk_ofs+3]:= Seg(my);
[ZNØ]   73: END;
[ØØØ]   74:
[MUØ]   75: BEGIN
[UP1]   76:   init_unit
[KNØ]   77: END.
```

Listing 1. Die Unit »stat.pas« muß in das zu analysierende Programm eingebunden werden

```
[UD3]    1: { Programm: ana.pas
[Z55]    2:   Funktion: Wertet das jeweilige
[B52]    3:    Programm aus
[EK3]    4:   Sprache: Turbo Pascal
[PD3]    5:   Autor: Peter Tiemann
[P35]    6:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    7:
[AW2]    8: PROGRAM analyse;
[ØØØ]    9:
[DWØ]   1Ø: {$A-}
[ØØØ]   11:
[5H1]   12: USES Crt;
[ØØØ]   13:
[CC3]   14: {$M 16384,5ØØØØ,65536Ø}
[ØØØ]   15:
[8Ø3]   16: TYPE elem = RECORD
[EE1]   17:               segment,
[792]   18:               offset : WORD;
[WU2]   19:               anzahl : LONGINT;
[5E2]   2Ø:               st     : POINTER;
[ZNØ]   21:             END;
[ØØØ]   22:
[SL6]   23: { maximal 15ØØØ Identifier verwaltbar }
[ØØØ]   24:
[1N2]   25: CONST max = 15ØØØ;
[ØØØ]   26:
[ØA6]   27: VAR tabelle    : ARRAY [1..max] OF ^elem;
[HG3]   28:     fuellstand : INTEGER;
[HR1]   29:     info_file,
[LT2]   3Ø:     map_file   : TEXT;
[884]   31:     trace_file : FILE OF WORD;
[1Ø1]   32:     zeile,
[8Y3]   33:     progname   : STRING[8Ø];
[AX1]   34:     bearbeitet,
[SF3]   35:     protosize  : LONGINT;
[KMØ]   36:     s,o,
[9B2]   37:     rel        : WORD;
[ØØØ]   38:
[9P5]   39: CONST ziffer : ARRAY[Ø..$F] OF CHAR
[AU2]   4Ø:              = 'Ø123456789ABCDEF';
[ØØØ]   41:
[V74]   42: TYPE string4 = STRING[4];
[ØØØ]   43:
[P77]   44: {  Die Routinen zur Umwandlung von Binär.-
[TØ6]   45:    in Hexdarstellung und umgekehrt. }
[ØØØ]   46:
[1Z5]   47: FUNCTION word2hex(n:WORD):string4;
[MUØ]   48: BEGIN
[LP2]   49:   word2hex[Ø]:= #4;
[G46]   5Ø:   word2hex[1]:= ziffer[ n SHR 12        ];
[NM6]   51:   word2hex[2]:= ziffer[(n SHR  8) AND $F];
[CM6]   52:   word2hex[3]:= ziffer[(n SHR  4) AND $F];
[H36]   53:   word2hex[4]:= ziffer[ n         AND $F];
[ZNØ]   54: END;
[ØØØ]   55:
[ØØØ]   56:
[976]   57: FUNCTION hex2word(h : string4) : WORD;
[4K4]   58: FUNCTION h2b(c : CHAR) : BYTE;
[MUØ]   59: BEGIN
[Y91]   6Ø:   CASE c OF
[B54]   61:     'Ø'..'9': h2b:= Ord(c)-Ord('Ø');
[2S4]   62:     'A'..'F': h2b:= Ord(c)-Ord('A')+1Ø;
[2S4]   63:     'a'..'f': h2b:= Ord(c)-Ord('a')+1Ø;
[8C2]   64:   ELSE      h2b:= Ø;
[ZNØ]   65:   END;
[ZNØ]   66: END;
[ØØØ]   67:
[MUØ]   68: BEGIN
[C83]   69:   hex2word:= h2b(h[4])
[M12]   7Ø:        + 16 * h2b(h[3])
[N52]   71:      + 256 * h2b(h[2])
[WL2]   72:     + 4Ø96 * h2b(h[1]);
[ZNØ]   73: END;
[ØØØ]   74:
[8S6]   75: { Diese Prozedur wird nach Öffnen einer
[EU6]   76:   Datei zum Lesen aufgerufen. Sie führt -
[WM5]   77:   z.B. falls die Datei nicht gefunden
[PF5]   78:   wurde - zum Programmabbruch.  }
[ØØØ]   79:
[NB5]   8Ø: PROCEDURE chk_io(name : STRING);
[MUØ]   81: BEGIN
[T53]   82:   IF IOResult<>Ø THEN
[MUØ]   83:     BEGIN
[BZ3]   84:       WriteLn('Fehler! ', name,
[713]   85:                 ' nicht gefunden!');
[BK3]   86:       WriteLn('- Abbruch. -');
[T41]   87:       Halt(1);
[ZNØ]   88:     END;
[ZNØ]   89: END;
[ØØØ]   9Ø:
[K77]   91: { Diese Prozedur öffnet die beiden Proto-
[W35]   92:   kolldateien sowie das Mapfile.
[TW6]   93:   Die Anzahl der zu bearbeitenden Einträge
[596]   94:   (= Anzahl der Prozedur- und Funktions-
[X24]   95:   aufrufe) wird errechnet.
[LY5]   96:   Die Verschiebung des Programmes im
[4P6]   97:   Speicher relativ zu den Segmentangaben
[ØU5]   98:   im Mapfile wird ermittelt, indem der
[2W6]   99:   Laufzeitwert des Segmentes der Unit STAT
[D96]  1ØØ:   mit dem Wert, der im Mapfile angegeben
[484]  1Ø1:   ist, verglichen wird. }
[ØØØ]  1Ø2:
[ZB6]  1Ø3: PROCEDURE eroeffne_eingabe_dateien;
[ØØØ]  1Ø4:
[4L3]  1Ø5: VAR st_map,st : WORD;
[ØØØ]  1Ø6:
[MUØ]  1Ø7: BEGIN
[M45]  1Ø8:   Assign(info_file, 'PROTO.INF');
[UZ2]  1Ø9:   Reset(info_file);
[SM6]  11Ø:   chk_io('Protokoll-Infofile PROTO.INF');
[EØ3]  111:   ReadLn(info_file);
[9M4]  112:   ReadLn(info_file, progname);
[F93]  113:   Dec(progname[Ø],3);
[GC5]  114:   Assign(map_file, progname+'MAP');
[XR2]  115:   Reset(map_file);
[CG5]  116:   chk_io('Mapfile '+progname+'MAP');
[1G4]  117:   Assign(trace_file, 'PROTO');
[333]  118:   Reset(trace_file);
[K55]  119:   chk_io('Protokollfile PROTO');
[SV6]  12Ø:   protosize:= FileSize(trace_file) DIV 2;
[VE2]  121:   bearbeitet:= Ø;
[PK2]  122:   fuellstand:= Ø;
[ØE6]  123:   WriteLn('Analysiere ',progname,'EXE');
[P31]  124:   REPEAT
[5Ø4]  125:     ReadLn(map_file, zeile);
[SM4]  126:   UNTIL Pos(' STAT ', zeile) <> Ø;
[2R5]  127:   st_map:= hex2word(Copy(zeile,2,4));
[EØ3]  128:   ReadLn(info_file);
[6L3]  129:   ReadLn(info_file, st);
[WV2]  13Ø:   rel:= st - st_map;
[HY2]  131:   Close(info_file);
[ZNØ]  132: END;
[ØØØ]  133:
[BX6]  134: { sucht im Mapfile nach der Prozedur oder
[3H6]  135:   Funktion, die bei Segment xs, Offset xo
[WD1]  136:   beginnt.
[WC6]  137:   Zuerst wird unter den 'PUBLICS' nachge-
[SV5]  138:   sehen, dann unter 'Line Numbers'. Im
[F96]  139:   zweiten Fall(dies kommt vor bei ver-
[847]  14Ø:   schachtelten Prozeduren oder Funktionen)
[2P6]  141:   kann nur mit dem Hinweis auf die Zeilen-
[PG6]  142:   nummer in einer Unit gedient werden. }
[ØØØ]  143:
[865]  144: PROCEDURE suche(xs, xo : WORD);
[ØØØ]  145:
[TK4]  146: VAR aktuell            : INTEGER;
[3D4]  147:     zeile,such,name    : STRING;
[ØØØ]  148:
[N46]  149: PROCEDURE suche_nach_zeilennummer;
[ØØØ]  15Ø:
[PZ5]  151: VAR numstr,unitname             : STRING;
[E66]  152:     zeilennummer,gelesen,p1,p2  : INTEGER;
[3H4]  153:     gefunden                    : BOOLEAN;
[ØØØ]  154:
[MUØ]  155: BEGIN
[KR2]  156:   gefunden:= FALSE;
[P31]  157:   REPEAT
[K36]  158:     IF Copy(zeile,1,12)='Line numbers' THEN
[MUØ]  159:       BEGIN
[H13]  16Ø:         p1:= Pos('(', zeile);
[S13]  161:         p2:= Pos(')', zeile);
[MH5]  162:         unitname:= Copy(zeile,p1+1,p2-p1-1)
[4MØ]  163:       END
[ZRØ]  164:     ELSE
[MUØ]  165:       BEGIN
[2E3]  166:         p1:= Pos(such, zeile);
[Z22]  167:         IF p1 <> Ø THEN
[MUØ]  168:           BEGIN
[3D1]  169:             Dec(p1,2);
[A41]  17Ø:             p2:= p1;
[EE4]  171:             WHILE zeile[p1] IN ['Ø'..'9']
[5K1]  172:               DO Dec(p1);
[F34]  173:             numstr:= Copy(zeile,p1+1,
[3Ø1]  174:                           p2-p1);
[1W4]  175:             Val(numstr,zeilennummer,p1);
[Z22]  176:             IF p1 <> Ø THEN
[MUØ]  177:               BEGIN
[PB3]  178:                 WriteLn('Fehler im',
[VS2]  179:                         ' Mapfileformat',
[PØ2]  18Ø:                         ' bei Zeile ',
[HM1]  181:                         zeile,'?');
[XV3]  182:                 name:= '?; Unleserliche'+
[CP2]  183:                         ' Zeilennummer '+
[A82]  184:                         'im Mapfile!';
[ØWØ]  185:                 Exit;
[ZNØ]  186:               END;
[5M2]  187:             gefunden:= TRUE;
[ZNØ]  188:           END;
[ZNØ]  189:       END;
[5Ø4]  19Ø:       ReadLn(map_file, zeile);
[M85]  191:   UNTIL gefunden OR EoF(map_file);
[UN2]  192:   IF gefunden THEN
[MUØ]  193:     BEGIN
[1E5]  194:       name:= 'Zeile '+numstr+' in Unit '+
[UK1]  195:              unitname;
[4MØ]  196:     END
[ZRØ]  197:   ELSE
[PM4]  198:     name:= '?; nicht bestimmbar!';
[ZNØ]  199: END;
[ØØØ]  2ØØ:
```

```
[ØØØ]  2Ø1:
[MUØ]  2Ø2: BEGIN
[NZ5]  2Ø3:   such:= word2hex(xs)+':'+word2hex(xo);
[KZ2]  2Ø4:   GotoXY(1, WhereY);
[9M2]  2Ø5:   Inc(bearbeitet);
[576]  2Ø6:   Write('(',bearbeitet*1ØØ/protosize :2:2,
[T52]  2Ø7:         '%) ',such,'.. ');
[TX1]  2Ø8:   aktuell:= 1;
[L15]  2Ø9:   WHILE aktuell <= fuellstand DO
[AD4]  21Ø:     WITH tabelle[aktuell]^ DO
[MUØ]  211:       BEGIN
[2C5]  212:         IF (segment = xs) AND (offset = xo)
[2RØ]  213:           THEN
[MUØ]  214:           BEGIN
[SØ2]  215:            Inc(anzahl);
[6N4]  216:            Write(anzahl,'.ter Aufruf    ');
[ØWØ]  217:            Exit;
[ZNØ]  218:           END;
[N72]  219:           Inc(aktuell);
[ZNØ]  22Ø:       END;
[EY3]  221:   IF fuellstand = max THEN
[MUØ]  222:     BEGIN
[9D5]  223:       WriteLn('Zu viele Prozeduren/ ',
[ØK2]  224:               'Funktionen!');
[416]  225:       WriteLn('Die Obergrenze ist ',max);
[Y55]  226:       WriteLn('Es werden nicht alle ',
[KA3]  227:               'berücksichtigt!');
[ZNØ]  228:     END;
[4S2]  229:   Inc(fuellstand);
[UG4]  23Ø:   New(tabelle[fuellstand]);
[RØ5]  231:   WITH tabelle[fuellstand]^ DO
[MUØ]  232:     BEGIN
[Z62]  233:       segment:= xs;
[522]  234:       offset:= xo;
[DV1]  235:       anzahl:= 1;
[Z44]  236:       Write('erster Aufruf    ');
[XR2]  237:       Reset(map_file);
[P31]  238:       REPEAT
[5Ø4]  239:         ReadLn(map_file, zeile);
[A93]  24Ø:       UNTIL EoF(map_file)
[G44]  241:         OR (Copy(zeile,2,9) = such)
[AA5]  242:         OR (Copy(zeile,1,9) = 'Line numb');
[CN4]  243:       IF Copy(zeile,2,9) = such THEN
[6T3]  244:         name:= Copy(zeile,18,255)
[ZRØ]  245:       ELSE
[MUØ]  246:         BEGIN
[H33]  247:           IF (Copy(zeile,1,12)=
[253]  248:               'Line numbers') THEN
[ZF4]  249:             suche_nach_zeilennummer
[ZRØ]  25Ø:           ELSE
[G74]  251:             name:= '?; Bitte MAP FILE: '+
[YS3]  252:                    'DETAILED; DEBUG INFO: '+
[NM2]  253:                    'ON einstellen!'
[ZNØ]  254:         END;
[W54]  255:       GetMem(st, Length(name)+1);
[Z63]  256:       STRING(st^):= name;
[ZNØ]  257:     END;
[ZNØ]  258: END;
[ØØØ]  259:
[DW6]  26Ø: {  Bubblesort-Verfahren zum Sortieren. }
[ØØØ]  261:
[XY4]  262: PROCEDURE sortiere_tabelle;
[ØØØ]  263:
[KF3]  264: VAR dummy : POINTER;
[B92]  265:     i     : INTEGER;
[ZS2]  266:     fertig: BOOLEAN;
[MUØ]  267: BEGIN
[P31]  268:   REPEAT
[RC2]  269:     fertig:= TRUE;
[7H4]  27Ø:     FOR i:= 1 TO fuellstand-1 DO
[MUØ]  271:       BEGIN
[7X3]  272:         IF tabelle[i]^.anzahl <
[F44]  273:           tabelle[i+1]^.anzahl THEN
[MUØ]  274:           BEGIN
[6K2]  275:             fertig:= FALSE;
[MD3]  276:             dummy:= tabelle[i];
[FD4]  277:             tabelle[i]:= tabelle[i+1];
[UA3]  278:             tabelle[i+1]:= dummy;
[ZNØ]  279:           END;
[ZNØ]  28Ø:       END;
[382]  281:   UNTIL fertig;
[ZNØ]  282: END;
[ØØØ]  283:
[ØØØ]  284:
[RK4]  285: PROCEDURE ausgabe_tabelle;
[ØØØ]  286:
[2P2]  287: VAR i : INTEGER;
[HD1]  288:     t : TEXT;
[MUØ]  289: BEGIN
[YZ2]  29Ø:   Assign(t, 'profil');
[UZ1]  291:   Rewrite(t);
[PR6]  292:   WriteLn(t,'Laufzeitprofil zu ',progname,
[R31]  293:           'EXE');
[DB1]  294:   ClrScr;
[HN5]  295:   Window(1,1,Lo(windmax),Hi(windmax));
[8W6]  296:   WriteLn('= Adresse = Aufrufe = Name ',
[F36]  297:          '==========');
[MN5]  298:   Window(1,2,Lo(windmax),Hi(windmax));
[EB4]  299:   FOR i:= 1 TO fuellstand DO
[HH3]  3ØØ:     WITH tabelle[i]^ DO
[MUØ]  3Ø1:       BEGIN
[A65]  3Ø2:         WriteLn(t,word2hex(segment),':',
[VD4]  3Ø3:                 word2hex(offset),anzahl:7,
[863]  3Ø4:                 '  ( ',STRING(st^),' )');
[GY4]  3Ø5:         WriteLn(word2hex(segment),':',
[VD4]  3Ø6:                 word2hex(offset),anzahl:7,
[863]  3Ø7:                 '  ( ',STRING(st^),' )');
[SD4]  3Ø8:         IF WhereY = Hi(windmax) THEN
[MUØ]  3Ø9:           BEGIN
[4P3]  31Ø:             Write('Bitte Taste..');
[P31]  311:             REPEAT
[N23]  312:             UNTIL ReadKey <> #Ø;
[DB1]  313:             ClrScr;
[ZNØ]  314:           END;
[ZNØ]  315:       END;
[AF1]  316:   Close(t);
[LK6]  317:   WriteLn('Das Profil finden Sie auch in ',
[833]  318:           'der Datei PROFIL.');
[ZNØ]  319: END;
[ØØØ]  32Ø:
[MUØ]  321: BEGIN
[SC4]  322:   eroeffne_eingabe_dateien;
[333]  323:   Reset(trace_file);
[P31]  324:   REPEAT
[BV3]  325:     Read(trace_file, s, o);
[MP1]  326:     Dec(s, rel);
[ZV1]  327:     suche(s, o);
[BP3]  328:   UNTIL EoF(trace_file);
[U13]  329:   Close(trace_file);
[MP2]  33Ø:   Close(map_file);
[P33]  331:   sortiere_tabelle;
[KU2]  332:   ausgabe_tabelle;
[KNØ]  333: END.
```

Listing 2. Mit »ana.pas« erfolgt die Auswertung des jeweiligen Programms

```
[AF3]    1: { Programm: demo.pas
[YA4]    2:   Funktion: Demo für ana.pas
[EK3]    3:   Sprache: Turbo Pascal
[PD3]    4:   Autor: Peter Tiemann
[P35]    5:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    6:
[492]    7: PROGRAM demo;
[ØØØ]    8:
[HL2]    9: USES Crt, stat;
[ØØØ]   1Ø:
[DK7]   11: {$S+  Stack Check muß eingeschaltet sein! }
[ØØØ]   12:
[ØØØ]   13:
[G25]   14: PROCEDURE ausgabe(e : BOOLEAN);
[ØØØ]   15:
[L82]   16: PROCEDURE tr;
[ØØØ]   17:
[Y22]   18: VAR a : WORD;
[ØØØ]   19:
[MUØ]   2Ø: BEGIN
[Y12]   21:   Write('True')
[ZNØ]   22: END;
[ØØØ]   23:
[A62]   24: PROCEDURE fa;
[ØØØ]   25:
[MUØ]   26: BEGIN
[Ø62]   27:   Write('False')
[ZNØ]   28: END;
[MUØ]   29: BEGIN
[6B1]   3Ø:   IF e THEN
[YCØ]   31:     tr
[ZRØ]   32:   ELSE
[FAØ]   33:     fa
[ZNØ]   34: END;
[ØØØ]   35:
[A23]   36: PROCEDURE eingabe;
[ØØØ]   37:
[MUØ]   38: BEGIN
[XF1]   39:   WriteLn;
[5V4]   4Ø:   Write('Bitte Taste drücken');
[XF1]   41:   WriteLn;
[P31]   42:   REPEAT
[N23]   43:   UNTIL ReadKey <> #Ø;
[ZNØ]   44: END;
[ØØØ]   45:
[MUØ]   46: BEGIN
[VB1]   47:   eingabe;
[KE2]   48:   ausgabe(TRUE);
[VB1]   49:   eingabe;
[VK2]   5Ø:   ausgabe(FALSE);
[ØA1]   51:   eingabe
[KNØ]   52: END.
```

Listing 3. »demo.pas« veranschaulicht die Funktionsweise des Analyseprogramms

MASM ab 4.0/TASM ab 1.0

Fehlersuche mit Registeranzeige

Wenn Assembler-Programme nicht die gewünschten Resultate zeigen, kann die Suche nach dem fehlerhaften Programmteil sehr zeitaufwendig werden. Mit dem DOS-Programm »debug« können Breakpoints mit Registerausgaben gesetzt werden. Wir stellen Ihnen hier eine praktische und elegante Alternative zur Verfügung.

Das wichtigste Werkzeug für einen Assembler-Programmierer auf Fehlersuche ist in der Regel ein Debugger wie beispielsweise das DOS-Programm »debug«. Dieses Programm ist sehr leistungsfähig, jedoch in der Handhabung relativ kompliziert und gewöhnungsbedürftig.
Das vorliegende Unterprogramm »WRITE_REGISTERS«, enthalten in der Datei »regwrite.asm« (Listing 1), bildet für bestimmte Programmtestzwecke eine einfache Alternative. »debug« läßt unter anderem mit Hilfe des Go-Befehls das Ausführen einer Com-Datei bis zu einer vom Anwender vorgegebenen Adresse zu. Hier die Syntax des Go-Befehls:

```
g [=adresse1] [adresse 2]
```

Die eckigen Klammern kennzeichnen die Adressen als optional. Wird »adresse 1« angegeben, beginnt die Programmausführung bei dieser Adresse, sonst am Programmanfang. Bei »adresse 2« wird, falls angegeben, das Programm unterbrochen und die aktuellen Registerinhalte angezeigt, sonst läuft das Programm bis zum Ende. »adresse 2« heißt auf Grund ihrer Aufgabe »Breakpoint«. »debug« unterscheidet die beiden Adressen durch das Gleichheitszeichen. Die Routine »WRITE_REGISTERS« ist in der Lage, diesen Befehl zu simulieren. Durch den Prozeduraufruf

```
call WRITE_REGISTERS
```

an einer beliebigen Stelle im Anweisungsteil eines im Com-Format gehaltenen Assembler-Programms, werden mit Ausnahme des Befehlszeigers (Instruction Pointers »IP«) die momentanen Registerinhalte auf dem Bildschirm ausgegeben. Die Zustände der einzelnen Flags sind dabei durch Binärzahlen und nicht durch Symbole wie »CY« oder »NC« im Fall des Carry-Flags dargestellt.
Nach der Bildschirmausgabe der Registerinhalte und Flags wartet das Unterprogramm auf einen beliebigen Tastendruck, um dann in das aufrufende Programm zurückzukehren. In der vorliegenden Form kann Regwrite mit dem Makro-Assembler MASM ab Version 4.0 oder TASM ab Version 1.0 assembliert werden und beispielsweise als Objektdatei in eine bereits bestehende Library miteinbezogen werden. Der Assemblier-Vorgang erfolgt durch

```
masm regwrite.asm
```

beziehungsweise durch

```
tasm regwrite.asm
```

Ein Programm, das auf »WRITE_REGISTERS« zurückgreift, muß diese Prozedur im Programmkopf als extern deklarieren. Dies geschieht mit der Direktive

```
extrn WRITE_REGISTERS: near
```

Um die Prozedur mit »call« aufrufen zu können, muß die Objektdatei »regwrite.obj« mit »Tlink« oder »Link« zum Hauptprogramm eingebunden werden. Den Umgang mit »WRITE_REGISTERS« soll das kurze Demoprogramm »register.asm« (Listing 2) verdeutlichen; die ablauffähige Version ist in Listing 3 dargestellt. Es belegt die allgemeinen Register »AX«, »BX«, »CX« und »DX« mit den Werten Null, Eins, Zwei , Drei und setzt sämtliche Flags mit Ausnahme des Trap-Flags auf Eins. Durch den Aufruf von »WRITE_REGISTERS« erscheinen diese Werte dann auf dem Bildschirm.
Durch das Einbinden von »regwrite.obj« sind Sie in der Lage, bequem an zahlreichen Stellen komplexerer Programme einen Status festzustellen, oder auch verschiedene Parameterauswertungen durchzutesten. Weiterhin kann durch DOS-Ausgabeumleitung, beispielsweise durch

```
register > test
```

ein externes Protokoll angelegt und erst nach Programmablauf ausgewertet werden.

(Klaus Joosz/bj)

```
[KØ5]    1: ; Programm: write_registers
[AT6]    2: ; Funktion: Diese Prozedur simuliert den
[D75]    3: ;  GO-Befehl des MS-DOS Debuggers
[HX1]    4: ;  debug.exe.
[NX2]    5: ; Sprache: Masm 5.Ø
[PB3]    6: ; Autor: Klaus Joosz
[FU4]    7: ; (c)1991 DMV Widuch GmbH & Co. KG
[ØØØ]    8:
[SG3]    9: code     SEGMENT PUBLIC
[L84]   1Ø:          public  WRITE_REGISTERS
[HB5]   11:          assume  cs:code, ds:code, ss:code
[HK2]   12:          assume  es:code
[ØØØ]   13:
[EL1]   14: regwrite:
[VU4]   15: write_registers proc    near
[ZXØ]   16:          pushf
[PF1]   17:          push    es
[KG1]   18:          push    ss
[MF1]   19:          push    ds
[KF1]   2Ø:          push    cs
[PE1]   21:          push    di
[MF1]   22:          push    si
[7F1]   23:          push    bp
[9G1]   24:          push    sp
[3G1]   25:          push    dx
[1G1]   26:          push    cx
[ZF1]   27:          push    bx
[XF1]   28:          push    ax
[DØ5]   29:          jmp     start_write_registers
[ØØØ]   3Ø:
[6Ø6]   31: register_string db    13,1Ø,32,32,32,"AX:"
[PT2]   32:                  db    32,32,32,32,32,"h"
[8R2]   33:                  db    32,32,32,32,"BX:"
```

```
[PT2]   34:                  db    32,32,32,32,32,"h"
[9R2]   35:                  db    32,32,32,32,"CX:"
[PT2]   36:                  db    32,32,32,32,32,"h"
[AR2]   37:                  db    32,32,32,32,"DX:"
[PT2]   38:                  db    32,32,32,32,32,"h"
[9R2]   39:                  db    32,32,32,32,"SP:"
[PT2]   4Ø:                  db    32,32,32,32,32,"h"
[SP2]   41:                  db    32,32,32,32,"BP:"
[PT2]   42:                  db    32,32,32,32,32,"h"
[UP2]   43:                  db    32,32,32,32,13,1Ø
[E23]   44:                  db    13,1Ø,32,32,32,"SI:"
[PT2]   45:                  db    32,32,32,32,32,"h"
[CP2]   46:                  db    32,32,32,32,"DI:"
[PT2]   47:                  db    32,32,32,32,32,"h"
[ZP2]   48:                  db    32,32,32,32,"CS:"
[PT2]   49:                  db    32,32,32,32,32,"h"
[ØR2]   5Ø:                  db    32,32,32,32,"DS:"
[PT2]   51:                  db    32,32,32,32,32,"h"
[FR2]   52:                  db    32,32,32,32,"SS:"
[PT2]   53:                  db    32,32,32,32,32,"h"
[1R2]   54:                  db    32,32,32,32,"ES:"
[PT2]   55:                  db    32,32,32,32,32,"h"
[C82]   56:                  db    13,1Ø,13,1Ø,"$"
[7K5]   57: flag_string      db    13,1Ø,32,32,32,"OF:"
[413]   58:                  db    32,32,32,32,32,"DF:"
[S13]   59:                  db    32,32,32,32,32,"IF:"
[433]   6Ø:                  db    32,32,32,32,32,"TF:"
[Ø33]   61:                  db    32,32,32,32,32,"SF:"
[W33]   62:                  db    32,32,32,32,32,"ZF:"
[SØ3]   63:                  db    32,32,32,32,32,"AF:"
[M23]   64:                  db    32,32,32,32,32,"PF:"
[Ø13]   65:                  db    32,32,32,32,32,"CF:"
[192]   66:                  db    32,32,13,1Ø,"$"
```

```
[6Z4]   67: flag_mask       dw    ØØØØ1ØØØØØØØØØØb
[PU2]   68:                 dw    ØØØØØ1ØØØØØØØØØb
[RU2]   69:                 dw    ØØØØØØ1ØØØØØØØØb
[SU2]   7Ø:                 dw    ØØØØØØØ1ØØØØØØØb
[TU2]   71:                 dw    ØØØØØØØØ1ØØØØØØb
[PU2]   72:                 dw    ØØØØØØØØØ1ØØØØØb
[SU2]   73:                 dw    ØØØØØØØØØØ1ØØØØb
[PU2]   74:                 dw    ØØØØØØØØØØØ1ØØØb
[SU2]   75:                 dw    ØØØØØØØØØØØØ1ØØb
[HP5]   76: wait_string     db    13,1Ø,"weiter -> "
[BD3]   77:                 db    "beliebige Taste$"
[ØØØ]   78:
[Ø54]   79: start_write_registers:
[PU1]   8Ø:         mov     bp,sp
[7K4]   81:         ; bp zeigt auf Stapelspitze
[KS1]   82:         mov     cx,12
[HM5]   83:         ; Schleifenzähler für 12 Register
[443]   84:         ; (ohne Flags und IP)
[KS1]   85:         mov     si,Ø9
[1Z4]   86:         ; Index von register_string
[K52]   87: start_loop:
[2C4]   88:         ; Beginn äußere Schleife
[YD2]   89:         mov     bx,[bp]
[Y55]   9Ø:         ; Wort vom Stapel nach bx über-
[H91]   91:         ; tragen
[L34]   92: check_if_stackpointer:
[BR1]   93:         cmp     cx,Ø8
[585]   94:         ; Wenn der Stack Pointer bearbei-
[R65]   95:         ; tet wird, steht der Schleifen-
[B92]   96:         ; zähler auf 8
[Ø62]   97:         jne     not_sp
[VM1]   98:         add     bx,18
[X55]   99:         ; Der Inhalt des Stack Pointers
[V56]  1ØØ:         ; muß so korrigiert werden, daß
[F85]  1Ø1:         ; er dem Wert vor Aufruf der Pro-
[RA6]  1Ø2:         ; zedur WRITE_REGISTERS entspricht
[PD1]  1Ø3: not_sp:
[1G1]  1Ø4:         push    cx
[396]  1Ø5:         ; Schleifenzähler äußere Schleife
[BE1]  1Ø6:         ; sichern
[HR1]  1Ø7:         mov     cl,Ø4
[RY3]  1Ø8:         ; Anzahl der Rotationen
[5R1]  1Ø9:         mov     ch,Ø4
[2R5]  11Ø:         ; Schleifenzähler innere Schleife
```

```
[962]  111: bin_to_hex:
[4Ø4]  112:         ; Beginn innere Schleife
[ST1]  113:         rol     bx,cl
[NT1]  114:         mov     dl,bl
[4G2]  115:         and     dl,ØØØØ1111b
[T55]  116:         ; Niedriges Halbbyte maskieren
[YL1]  117:         add     dl,48
[EG4]  118:         ; Umwandlung Zahlenwert ->
[7X1]  119:         ; ASCII-Code
[EP1]  12Ø:         cmp     dl,58
[CT2]  121:         jb      put_string
[SL1]  122:         add     dl,Ø7
[L26]  123:         ; Falls ein Zahlenwert größer 9
[CB5]  124:         ; dargestellt werden soll, muß
[MV4]  125:         ; im Hexadezimalsystem auf die
[BM4]  126:         ; Buchstaben A,..,F zurückge-
[ML2]  127:         ; griffen werden
[U62]  128: put_string:
[S75]  129:         mov     register_string[si],dl
[BF5]  13Ø:         ; ASCII-Code in register_string
[9Z1]  131:         ; eintragen
[9B1]  132:         inc     si
[1D3]  133:         ; si inkrementieren
[191]  134:         dec     ch
[1P5]  135:         ; Schleifenzähler dekrementieren
[713]  136:         jnz     bin_to_hex
[315]  137:         ; Falls ch ungleich Ø -> weitere
[HM1]  138:         ; Schleife
[HM1]  139:         add     si,Ø9
[6N5]  14Ø:         ; Nächste Hex-Ziffer in register_
[MB1]  141:         ; string
[4R1]  142:         cmp     si,87
[CT5]  143:         ; Falls Index gleich 87, so beginnt
[8R5]  144:         ; eine neue Zeile -> größerer Zwi-
[3Ø2]  145:         ; schenraum
[EL1]  146:         je      add_7
[473]  147:         jmp     no_addition
[TYØ]  148: add_7:
[DM1]  149:         add     si,Ø7
[382]  15Ø: no_addition:
```

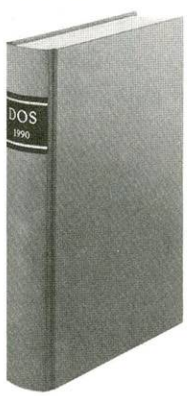
Listing 1. »regwrite.asm« enthält die Prozedur »WRITE REGISTERS«, mit deren Hilfe ein programmkontrolliertes Debugging durchführbar wird

Wer Wo Wann?

Bei Zuschriften bitten wir, folgendes zu beachten: Haben Sie Fragen, Anregungen oder Programme, die Sie der DOS-Redaktion zukommen lassen möchten, adressieren Sie wie folgt:

Redaktion
»DOS International«
Gruberstraße 46c
8011 Poing
Tel.: (0 81 21) 7 69-0
Fax: (0 81 21) 7 90 46

Ein schönes Buch
binden wir aus Ihrem
DOS



Wenn Sie einen Jahrgang DOS an unser Postfach schicken, erhalten Sie 4 Wochen später ein gut gebundenes Buch zurück. Es hat einen blauen Einband, eine Goldprägung auf dem Rücken und kostet DM 35,80 pro Band. Ältere Jahrgänge binden wir auch. Es sind sämtliche Einbanddecken vorrätig. Außerdem erledigen wir alle Buchbinderarbeiten vom individuell gestalteten Gästebuch oder Fotoalbum bis zum feinsten Lederband für Ihre Bibliothek.
Unsere Dokumentation gibt auf die anstehenden Fragen Antwort. Wir schicken sie Ihnen gerne.

Manfred Mayer
Sortimentsbuchbinderei GmbH
Postfach 10 31 31, Hasenbergstraße 95 A
7000 Stuttgart 1
Telefon (07 11) 6 36 62 85, Fax 63 41 92



PD-SERVICE LAGE
Postfach 17 43 * 4937 Lage/Lippe
FAX 0 52 32/40 39 * BTX *PD Service #

liefert die allerneueste
PD & SHAREWARE
für IBM-kompatible PCs!

Zudem haben wir eine der weltweit größten UNIX-Shareware-Bibliotheken!

Wir bieten Ihnen eine Riesenauswahl an VGA-GIF-Bildern, Spielen, DTP-Ergänzungen, dt. & internationalen Programmen.

Unsere Kopiergebühren:
3,00 - 4,50 DM für 5,25 Zoll & 5,00 - 6,50 DM für 3,5 Zoll
Katalogdisketten für 2,50 DM (Porto)
Mit uns sind Sie immer Up-to-date!

```
[PL1]  151:          add     bp,Ø2
[XB5]  152:          ; bp zeigt auf nächstes Wort auf
[RW1]  153:          ; dem Stapel
[VD1]  154:          pop     cx
[HF5]  155:          ; Schleifenzähler für äußere
[XH2]  156:          ; Schleife holen
[VØ3]  157:          loop    start_loop
[6T4]  158:          ; Ende der äußeren Schleife
[1F2]  159: examine_flags:
[YD2]  16Ø:          mov     bx,[bp]
[2M5]  161:          ; Flag-Register vom Stapel nach bx
[VA2]  162:          ; übertragen
[ZS1]  163:          mov     cx,Ø9
[BX4]  164:          ; Schleifenzähler für 9 Flags
[KS1]  165:          mov     si,Ø9
[W54]  166:          ; Index von flag_string
[ET1]  167:          xor     di,di
[LS4]  168:          ; Index von mask auf Ø setzen
[BØ2]  169: check_flag:
[HV1]  17Ø:          mov     dx,bx
[FT3]  171:          and     dx,flag_mask[di]
[1B4]  172:          ; einzelnes Bit maskieren
[ØV3]  173:          cmp     dx,flag_mask[di]
[CG3]  174:          ; war das Bit gesetzt?
[MD2]  175:          jne     no_flag
[6E4]  176:          mov     flag_string[si],"1"
[NF2]  177:          jmp     flag_set
[WF1]  178: no_flag:
[3E4]  179:          mov     flag_string[si],"Ø"
[EK1]  18Ø: flag_set:
[5L1]  181:          add     di,Ø2
[TP5]  182:          ; di zeigt auf nächstes Element in
[KYØ]  183:          ; mask
[FM1]  184:          add     si,Ø8
[1U5]  185:          ; si zeigt auf Platz für nächste
[6S4]  186:          ; Hex-Ziffer in flag_string
[MV2]  187:          loop    check_flag
[M74]  188: write_register_string:
[8K5]  189:          mov     dx,offset register_string
[3Y1]  19Ø:          mov     ah,Ø9h
[3K1]  191:          int     21h
[TC3]  192: write_flag_string:
[DR4]  193:          mov     dx,offset flag_string
[3Y1]  194:          mov     ah,Ø9h
[3K1]  195:          int     21h
[9G3]  196: write_wait_string:
[XU4]  197:          mov     dx,offset wait_string
[3Y1]  198:          mov     ah,Ø9h
[3K1]  199:          int     21h
[VH1]  2ØØ: read_key:
[ZX1]  2Ø1:          mov     ah,Ø7h
[DM4]  2Ø2:          ; Eingabe ohne Bildschirmecho
[3K1]  2Ø3:          int     21h
[CW3]  2Ø4: get_cursor_position:
[ZP1]  2Ø5:          mov     ah,Ø3
[CR5]  2Ø6:          ; aktuelle Cursor-Position bestim-
[BRØ]  2Ø7:          ; men
[VP1]  2Ø8:          mov     bh,ØØ
[PR3]  2Ø9:          ; auf Bildschirmseite Ø
[YH1]  21Ø:          int     1Øh
[NA5]  211:          ; Rückgabe der Cursor-Position:
[BG3]  212:          ;  dh: Zeile, dl: Spalte
[K63]  213: move_cursor_left:
[XP1]  214:          mov     ah,Ø2
[R25]  215:          ; Cursor-Position verändern
[BR1]  216:          mov     dl,ØØ
[424]  217:          ; Cursor an Zeilenanfang
[VP1]  218:          mov     bh,ØØ
[H53]  219:          ; Bildschirmseite Ø
[YH1]  22Ø:          int     1Øh
[C22]  221: clear_line:
[5R1]  222:          mov     ah,Ø6
[615]  223:          ; Löschvorgang initialisieren
[MT1]  224:          xor     al,al
[UW2]  225:          ; Fenster löschen
[1T1]  226:          mov     ch,dh
[175]  227:          ; Zeilennummer obere linke Ecke
[NT1]  228:          mov     cl,dl
[7G5]  229:          ; Spaltennummer obere linke Ecke
[4S1]  23Ø:          mov     dl,79
[ØU5]  231:          ; Spaltennummer untere rechte Ecke
[9R1]  232:          mov     bh,Ø7
[GB5]  233:          ; Anzeigeattribut der Leerzeile
[YH1]  234:          int     1Øh
[2U3]  235: end_write_registers:
[RD1]  236:          pop     ax
[TD1]  237:          pop     bx
[VD1]  238:          pop     cx
[XD1]  239:          pop     dx
[3E1]  24Ø:          pop     sp
[1D1]  241:          pop     bp
[ED1]  242:          pop     si
[GC1]  243:          pop     di
[ED1]  244:          pop     ds
[N53]  245:          ; Holt cs vom Stapel
[ED1]  246:          pop     ds
[CE1]  247:          pop     ss
[GD1]  248:          pop     es
[LSØ]  249:          popf
[3PØ]  25Ø:          ret
[TS3]  251: write_registers endp
[ØØØ]  252:
[TT1]  253: code     ends
[YE2]  254:          end     regwrite
```

Listing 1. »regwrite.asm« enthält die Prozedur »WRITE_REGISTERS«, mit deren Hilfe ein programmkontrolliertes Debugging durchführbar wird

```
[544]    1: ; Programm: register.asm
[GY6]    2: ; Funktion: Dieses Programm demonstriert
[PH4]    3: ;  das Einbinden der Prozedur
[DØ7]    4: ;  WRITE_REGISTERS, enthalten in der Datei
[N47]    5: ;  REGWRITE.ASM, in ein Assemblerprogramm.
[9G6]    6: ;  Die vier allgemeinen Register AX, BX,
[NX5]    7: ;  CX, DX werden mit den Werten Ø, 1, 2, 3
[6B6]    8: ;  belegt und mit Ausnahme des Trap-Flags
[LA4]    9: ;  alle Flags auf 1 gesetzt.
[SV6]   1Ø: ;  Vorgehensweise zum Erstellen der Datei
[8K4]   11: ;  register.com mit masm 5.Ø:
[HS2]   12: ;  - masm register
[KE4]   13: ;  - link register+regwrite
[U66]   14: ;  (Voraussetzung: Datei regwrite.obj
[4Ø7]   15: ;  befindet sich im aktuellen Verzeichnis)
[YG5]   16: ;  - exe2bin register register.com
[NW2]   17: ; Sprache: Masm %.Ø
[PB3]   18: ; Autor: Klaus Joosz
[FU4]   19: ; (c)1991 DMV Widuch GmbH & Co. KG
[ØØØ]   2Ø:
[K25]   21:          EXTRN   write_registers: near
[SG3]   22: code     SEGMENT PUBLIC
[R85]   23:          assume  cs:code,ds:code,ss:code
[HK2]   24:          assume  es:code
[AG1]   25:          org     1ØØh
[3UØ]   26: main:
[VX1]   27:          jmp     start
[5G5]   28: end_string       db      13,1Ø,"Wieder im "
[MF3]   29:                  db      "Hauptprogramm!"
[YP1]   3Ø:                  db      13,1Ø,"$"
[E21]   31: start:
[EK3]   32:          mov     ax,ØØØØ111Ø11Ø1Ø1Ø1b
[XF1]   33:          push    ax
[LSØ]   34:          popf
[Ø26]   35: ; Alle Flags bis auf Trap-Flag gesetzt
[ØØØ]   36:
[152]   37:          mov     ax,ØØØØh
[752]   38:          mov     bx,ØØØ1h
[D52]   39:          mov     cx,ØØØ2h
[L52]   4Ø:          mov     dx,ØØØ3h
[ØØØ]   41:
[A14]   42:          call    WRITE_REGISTERS
[ØØØ]   43:
[AL4]   44:          mov     dx,offset end_string
[3Y1]   45:          mov     ah,Ø9h
[3K1]   46:          int     21h
[6H1]   47: end_main:
[VY1]   48:          mov     ah,4ch
[3K1]   49:          int     21h
[ZP1]   5Ø: code     ends
[LN1]   51:          end     main
```

Listing 2. »register.asm« dient als Demonstrationsprogramm für die Debugging-Hilfe

```
[1K5]    1: EB 1E 9Ø ØD ØA 57 69 65 64 65 72 2Ø 69 6D 2Ø 48
[PD5]    2: 61 75 7Ø 74 7Ø 72 6F 67 72 61 6D 6D 21 ØD ØA 24
[FF5]    3: B8 D5 ØE 5Ø 9D B8 ØØ ØØ BB Ø1 ØØ B9 Ø2 ØØ BA Ø3
[6F5]    4: ØØ E8 ØC ØØ BA Ø3 Ø1 B4 Ø9 CD 21 B4 4C CD 21 ØØ
[RD5]    5: 9C Ø6 16 1E ØE 57 56 55 54 52 51 53 5Ø E9 24 Ø1
[V75]    6: ØD ØA 2Ø 2Ø 2Ø 41 58 3A 2Ø 2Ø 2Ø 2Ø 2Ø 68 2Ø 2Ø
[265]    7: 2Ø 2Ø 42 58 3A 2Ø 2Ø 2Ø 2Ø 2Ø 68 2Ø 2Ø 2Ø 2Ø 43
[N65]    8: 58 3A 2Ø 2Ø 2Ø 2Ø 2Ø 68 2Ø 2Ø 2Ø 2Ø 44 58 3A 2Ø
[C55]    9: 2Ø 2Ø 2Ø 2Ø 68 2Ø 2Ø 2Ø 2Ø 53 5Ø 3A 2Ø 2Ø 2Ø 2Ø
[455]   1Ø: 2Ø 68 2Ø 2Ø 2Ø 2Ø 42 5Ø 3A 2Ø 2Ø 2Ø 2Ø 2Ø 68 2Ø
[6A5]   11: 2Ø 2Ø 2Ø ØD ØA ØD ØA 2Ø 2Ø 2Ø 53 49 3A 2Ø 2Ø 2Ø
[C65]   12: 2Ø 2Ø 68 2Ø 2Ø 2Ø 2Ø 44 49 3A 2Ø 2Ø 2Ø 2Ø 2Ø 68
[V35]   13: 2Ø 2Ø 2Ø 2Ø 43 53 3A 2Ø 2Ø 2Ø 2Ø 2Ø 68 2Ø 2Ø 2Ø
[865]   14: 2Ø 44 53 3A 2Ø 2Ø 2Ø 2Ø 68 2Ø 2Ø 2Ø 2Ø 53 53
[965]   15: 3A 2Ø 2Ø 2Ø 2Ø 2Ø 68 2Ø 2Ø 2Ø 2Ø 45 53 3A 2Ø 2Ø
[NB5]   16: 2Ø 2Ø 2Ø 68 ØD ØA ØD ØA 24 ØD ØA 2Ø 2Ø 2Ø 4F 46
[L75]   17: 3A 2Ø 2Ø 2Ø 2Ø 2Ø 44 46 3A 2Ø 2Ø 2Ø 2Ø 2Ø 49 46
[N75]   18: 3A 2Ø 2Ø 2Ø 2Ø 2Ø 54 46 3A 2Ø 2Ø 2Ø 2Ø 2Ø 53 46
[W75]   19: 3A 2Ø 2Ø 2Ø 2Ø 2Ø 5A 46 3A 2Ø 2Ø 2Ø 2Ø 2Ø 41 46
[D75]   2Ø: 3A 2Ø 2Ø 2Ø 2Ø 2Ø 5Ø 46 3A 2Ø 2Ø 2Ø 2Ø 2Ø 43 46
[285]   21: 3A 2Ø 2Ø ØD ØA 24 ØØ Ø8 ØØ Ø4 ØØ Ø2 ØØ Ø1 8Ø ØØ
[G85]   22: 4Ø ØØ 1Ø ØØ Ø4 ØØ Ø1 ØØ ØD ØA 77 65 69 74 65 72
[RB5]   23: 2Ø 2D 3E 2Ø 62 65 6C 69 65 62 69 67 65 2Ø 54 61
[NH5]   24: 73 74 65 24 8B EC B9 ØC ØØ BE Ø9 ØØ 8B 5E ØØ 83
[YF5]   25: F9 Ø8 75 Ø3 83 C3 12 51 B1 Ø4 B5 Ø4 D3 C3 8A D3
[SF5]   26: 8Ø E2 ØF 8Ø C2 3Ø 8Ø FA 3A 72 Ø3 8Ø C2 Ø7 88 94
[3U5]   27: 5Ø Ø1 46 FE CD 75 E5 83 C6 Ø9 83 FE 57 74 Ø3 EB
[EM5]   28: Ø4 9Ø 83 C6 Ø7 83 C5 Ø2 59 E2 C1 8B 5E ØØ B9 Ø9
[VC5]   29: ØØ BE Ø9 ØØ 33 FF 8B D3 23 95 46 Ø2 3B 95 46 Ø2
[KL5]   3Ø: 75 Ø8 C6 84 F9 Ø1 31 EB Ø6 9Ø C6 84 F9 Ø1 3Ø 83
[5U5]   31: C7 Ø2 83 C6 Ø8 E2 DF BA 5Ø Ø1 B4 Ø9 CD 21 BA F9
[2K5]   32: Ø1 B4 Ø9 CD 21 BA 58 Ø2 B4 Ø9 CD 21 B4 Ø7 CD 21
[WK5]   33: B4 Ø3 B7 ØØ CD 1Ø B4 Ø2 B2 ØØ B7 ØØ CD 1Ø B4 Ø6
[EU5]   34: 32 CØ 8A EE 8A CA B2 4F B7 Ø7 CD 1Ø 58 5B 59 5A
[ZF3]   35: 5C 5D 5E 5F 1F 1F 17 Ø7 9D C3
```

Listing 3. In das ablauffähige Programm »register.com« ist die Debugging-Hilfe durch »link« integriert

NOVELL®

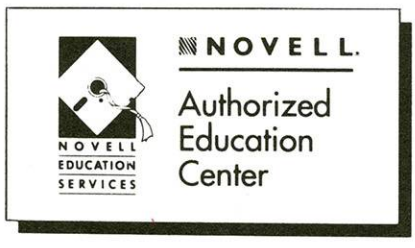
vertraut uns: Sie können es auch!

NOVELL NetWare®

Schulungen vom Profi!

Fordern Sie unseren aktuellen Schulungskatalog an!

Schulungen für Netzwerkadministration und -wartung, Installation und Anwendung von NOVELL NetWare®, NetWare® Communication Products, Microsoft® Produkte



LAN Schulungszentrum Schaeffer
NOVELL Authorized Education Center
Fokko-Ukena-Str.18, 2950 Leer
☎0491/12383 FAX.0491/13404

MASM und Quick-C

Der enträtselte Strichcode

Jeder kennt sie, die dunklen Striche auf den Lebensmittelverpackungen in den Supermärkten, und mancher hat sich gewundert, daß dieses scheinbar sinnlose Gewirr von unterschiedlich breiten Strichen offenbar doch einen Sinn ergibt.

Daß wirklich helle Köpfe hinter der Entwicklung des Strichcodes stecken, erkennt man spätestens daran, daß egal wie man die Scan-Vorlage hält – normal oder auf dem Kopf stehend – das Scanning erfolgreich arbeitet. Ja sogar Winkel bis zu 45 Grad sind möglich. Bei Lesestiften ist es auch gleichgültig, ob man mit dem Stift von rechts nach links oder von links nach rechts über den Strichcode fährt – es wird immer der gleiche Code erkannt.

Die Entwickler des EAN-Codes strebten eine Form zu finden, die lage- und richtungsunabhängiges Lesen ermöglicht

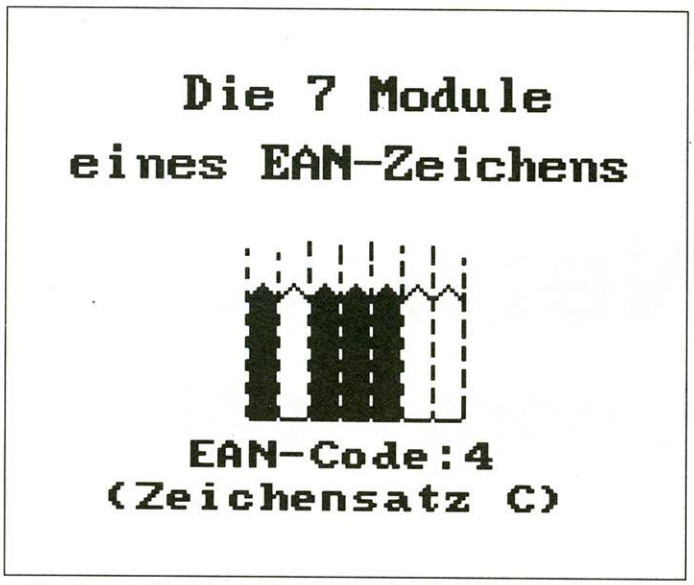


Bild 1. Ein EAN-Code-Zeichen genau betrachtet

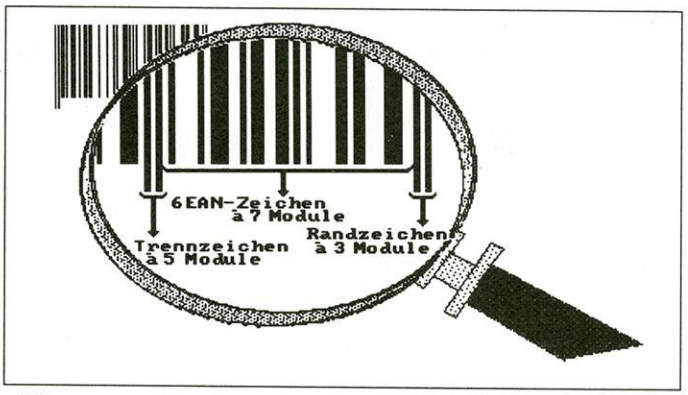


Bild 2. Der Aufbau des EAN-Codes unter die Lupe genommen

– sie haben ihr Ziel erreicht. Vor dem EAN-Symbol gab es in den USA und in Kanada bereits den UPC (Universal-Product-Code), auf den die EAN-Technologie aufbaut. EAN und UPC sind zueinander – zumindest theoretisch – kompatibel. Diese theoretische Kompatibilität begründet sich aus der Tatsache, daß amerikanische Scanner unseren 13stelligen Code nur zwölfstellig interpretieren. Bei genauerer Betrachtung des 13stelligen EAN-Strichcodes bemerkt man, daß hier nur zwölf Stellen codiert sind. Der UPC ist ebenfalls ein zwölfstelliger Code. Die 13. Stelle beim EAN-Code ist durch Verschlüsselung der sechs Ziffern in der linken Symbolhälfte dargestellt (die EAN-Ziffern sind von rechts nach links, von 1 bis 13 durchnumeriert).

Sollte es bis hierher schon etwas verwirrend erscheinen, so kommt es leider noch schlimmer.

Es gibt nämlich nicht nur 13stelligen, sondern auch achtstelligen EAN-Code, sechsstelligen UPC-E-Code, Strichcodierung für hausinterne Artikelnumerierungen und die Strichcodierung für Presseerzeugnisse. Aber lassen Sie sich nicht entmutigen, gegen Ende dieser Erörterungen sollte sich einiges klären.

Zerlegung des Strichcodes

Betrachten Sie sich zunächst den 13stelligen EAN-Code etwas genauer: Ein EAN-Zeichen besteht aus unterschiedlich dicken Balken und Zwischenräumen. Ein Balken oder Zwischenraum wird aus unterschiedlich vielen Modulen zusammengesetzt. Ein Modul ist wiederum das Rastermaß für die Breite von Balken und Zwischenräumen. Ein EAN-Zeichen wird nun aus sieben Modulen gebildet (Bild 1). Die Balken werden im folgenden als »1«, die Zwischenräume als »0« dargestellt. Von rechts nach links besteht der Code, wie in Bild 2 dargestellt, aus:

- einem Randzeichen, bestehend aus drei Modulen (codiert: 101),
- sechs Nutzzeichen, bestehend aus je sieben Modulen

Zeichen	Zeichensatz A	Zeichen	Zeichensatz B	Zeichen	Zeichensatz C
0	0001101	0	0100111	0	1110010
1	0011001	1	0110011	1	1100110
2	0010011	2	0011011	2	1101100
3	0111101	3	0100001	3	1000010
4	0100011	4	0011101	4	1011100
5	0110001	5	0111001	5	1001110
6	0101111	6	0000101	6	1010000
7	0111011	7	0010001	7	1000100
8	0110111	8	0001001	8	1001000
9	0001011	9	0010111	9	1110100

Tabelle 1. Die Zeichensätze A, B und C des EAN-Codes

(Stelle 1-6) aus dem Zeichensatz C (codiert: siehe Tabelle 1),
- einem Trennzeichen, bestehend aus fünf Modulen (codiert: 01010),
- sechs Nutzzeichen, bestehend aus je sieben Modulen (Stelle 7-12) aus den Zeichensätzen A und B (codiert: siehe Tabelle 1),
- und einem Randzeichen, bestehend aus drei Modulen (codiert: 101).

Die Stellen 7-12 werden aus einer bestimmten Abfolge der Zeichen aus den Zeichensätzen A und B gebildet. Die 13. Stelle wird aus dieser Abfolge codiert (siehe Tabelle 2).
Um die »Verwirrung« perfekt zu machen, kommen bei der Codierung der Nutzzeichen drei verschiedene Zeichensätze zum Einsatz.
Werden also zum Beispiel alle sechs Ziffern der Stellen 7-12 aus dem Zeichensatz A codiert, so besitzt die 13. Stelle den Wert »0«.
Noch ein Beispiel (für 90 12345 12345 1):

Stelle:	13	12	11	10	9	8	7	6	5	4	3	2	1
Zeichensatz:		A	B	B	A	B	A	C	C	C	C	C	C
Ziffer:	9	0	1	2	3	4	5	1	2	3	4	5	1

Die Stelle 1 ist gleichzeitig eine Prüfziffer.
Der achtstellige EAN-Code besteht aus:
- einem Randzeichen,
- vier Nutzzeichen (Zeichensatz C) in der rechten Hälfte,
- einem Trennzeichen,
- vier Nutzzeichen (Zeichensatz A) in der rechten Hälfte,
- und einem Randzeichen.

Beispiel (für die EAN 90 12345 0):

Stelle:	8	7	6	5	4	3	2	1
Zeichensatz:	A	A	A	A	C	C	C	C
Ziffer:	9	0	1	2	3	4	5	0

Der zwölfstellige UPC besteht aus:
- einem Randzeichen,
- sechs Nutzzeichen (Zeichensatz C) in der rechten Hälfte,
- einem Trennzeichen,
- vier Nutzzeichen (Zeichensatz A) in der rechten Hälfte,
- und einem Randzeichen.

Beispiel (für den UPC (0) 12345 67890 5):

Stelle:	12	11	10	9	8	7	6	5	4	3	2	1
Zeichensatz:	A	A	A	A	A	C	C	C	C	C	C	C
Ziffer:	0	1	2	3	4	5	6	7	8	9	0	5

Soweitdie wichtigsten Strichcodes. Mit den hier aufgestellten Informationen sowie den Programmbeispielen ausgestattet, sollte es möglich sein, Strichcodes zu erzeugen und zu drucken.

Tabelle	Zeichensatzfolge (Stelle 7-12)
0	A A A A A A
1	A A B A B B
2	A A B B A B
3	A A B B B A
4	A B A A B B
5	A B B A A B
6	A B B B A A
7	A B A B A B
8	A B A B B A
9	A B B A B A

Tabelle 2. Das 13. Zeichen wird aus einer Abfolge der Zeichensätze A und B aus den Stellen 7 bis 12 gebildet

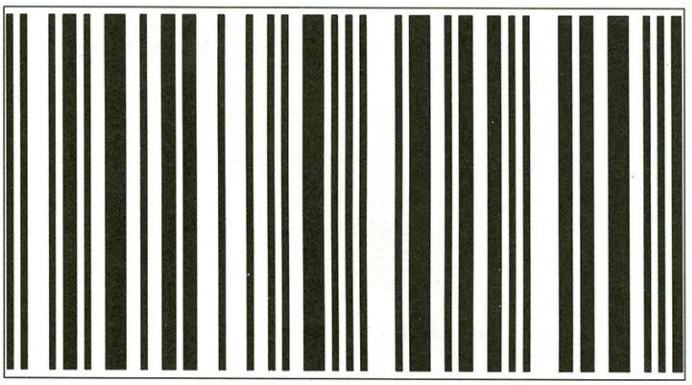


Bild 3. Die Ausgabe des Programms »eandemo« anhand der »Standard-EAN« (49 02580 30162 0)

Programme zum Lesen, Darstellen und Drucken der Strichcodes

Als Beispiel sollte das Programm »eandemo.c« (Listing 1) gute Dienste leisten. Es verlangt als Parameter eine 13stellige EAN und druckt den richtigen Code auf IBM-Proprinter- sowie Epson-kompatiblen 9-Nadel-Druckern aus. Gleichzeitig erfolgt eine Darstellung auf einem VGA-Bildschirm. Das Druckerausgabeergebnis konnte der mir zur Verfügung stehende Handscanner (Datalogic Dl 60) problemlos erkennen. Wird beim Start kein Parameter angegeben, so wird die Standard-EAN (49 02580 30162 0) angenommen (Bild 3). Diese EAN befindet sich auf den Disketten-boxen der Marke Maxell MD2-HD.
Besonders interessant für Programmierer oder Besitzer eines Barcode-Scanners oder Lesestifts sind die Programme »scan.asm« (MASM ab Version 3.0, Listing 2) und »readean.c« (Listing 3), die Sie zusammenlinken müssen. Sofern Sie zum Linken lediglich die Objektdatei von »scan« benötigen, finden Sie diese in Listing 4. »scan.asm« stellt für »readean.c« eine Interrupt-Serviceroutine zur Verfügung, um die mit »eandemo.c« ausgedruckten Codes (oder Codes auf Verpackungen) einlesen zu können.

(Josef Eder/bj)

```
[2F3]    1: /* Programm: eandemo.c
[MA6]    2:    Funktion: Demonstration von readean
[MK2]    3:    Srache: Quick C
[NT2]    4:    Autor: Josef Eder
[T15]    5:    (c)1991 DMV Widuch GmbH & Co. KG */
[ØØØ]    6:
[L13]    7: #include <stdio.h>
[MZ2]    8: #include <graph.h>
[ØØØ]    9:
[UH4]   1Ø: #define SPALTEN 4*(12*7+2*3+5+7)
[213]   11: /*                 ^    ^ ^ ^  ^ ^
[CD2]   12:                    |    | | |  | |
[YU4]   13:         4Striche/Modul  | |  | |
[TD3]   14:                   12Ziffern |  | |
[ZK3]   15:              7Module/Ziffer |  | |
[4F4]   16:          2Randzeichen à 3Module | |
[TM4]   17:           1Trennzeichen à 5Module |
[AZ5]   18:      7Striche Platz am Beginn fuer 1. Zahl
[17Ø]   19: */
[ØØØ]   2Ø:
[TV2]   21: #define ESC     Øx1b
[ØØØ]   22:
[A57]   23: /* Werte für veschiedene Grafikdichten   */
```

```
[LS5]   24: /* für 8-Nadel Grafikdrucker              */
[236]   25: /* zb. EPSON oder IBM-PROPRINTER          */
[ØØØ]   26:
[BU5]   27: #define MØ Ø /* einfache Dichte          */
[4Y5]   28: #define M1 1 /* doppelte Dichte          */
[MP6]   29: #define M2 2 /* doppelte Dichte ,schnell */
[RU5]   3Ø: #define M3 3 /* vierfache Dichte         */
[T76]   31: #define M4 4 /* Bildschirmgrafiken I     */
[LA6]   32: #define M6 6 /* Bildschirmgrafiken II    */
[ØØØ]   33:
[ØØØ]   34:
[TT4]   35: typedef BYTE unsigned char;
[ØØØ]   36:
[RS2]   37: BYTE ean_null=Ø;
[ØØØ]   38:
[ZB6]   39: /* RANDZEICHEN besteht aus 3 MODULEN     */
[EM5]   4Ø: BYTE eanrand[3]="\xff\xØØ\xff";
[VU6]   41: /* TRENNZEICHEN besteht aus 5 MODULEN    */
[UV6]   42: BYTE eantrenn[5]="\xØØ\xff\xØØ\xff\xØØ";
[ØØØ]   43:
[397]   44: /* eanparity[i][j]: zur Bestimmung  der  */
```

Listing 1. »eandemo.c« demonstriert die EAN-Code-Ausgabe

```
[GA6]   45: /* Abfolge von Ziffern aus den Zeichen-  */
[DD6]   46: /* sätzten A od. B (Stelle 7-12 der EAN) */
[L24]   47: BYTE eanparity[10][6]=
[P70]   48:       {
[M62]   49:       { 1,1,1,1,1,1 },
[Y62]   50:       { 1,1,2,1,2,2 },
[W62]   51:       { 1,1,2,2,1,2 },
[Y62]   52:       { 1,1,2,2,2,1 },
[W62]   53:       { 1,2,1,1,2,2 },
[W62]   54:       { 1,2,2,1,1,2 },
[W62]   55:       { 1,2,2,2,1,1 },
[U62]   56:       { 1,2,1,2,1,2 },
[W62]   57:       { 1,2,1,2,2,1 },
[K52]   58:       { 1,2,2,1,2,1 }
[BD0]   59:       };
[000]   60:
[K36]   61: /* eancode1[][]: Zeichensatz A           */
[KH3]   62: BYTE eancode1[10][7]=
[P70]   63:       {
[2X6]   64: /* 0 */     "\x00\x00\x00\xff\xff\x00\xff",
[3X6]   65: /* 1 */     "\x00\x00\xff\xff\x00\x00\xff",
[4X6]   66: /* 2 */     "\x00\x00\xff\x00\x00\xff\xff",
[917]   67: /* 3 */     "\x00\xff\xff\xff\x00\x00\xff",
[6X6]   68: /* 4 */     "\x00\xff\x00\x00\x00\xff\xff",
[7X6]   69: /* 5 */     "\x00\xff\xff\x00\x00\x00\xff",
[C17]   70: /* 6 */     "\x00\xff\x00\xff\xff\xff\xff",
[D17]   71: /* 7 */     "\x00\xff\xff\xff\x00\xff\xff",
[E17]   72: /* 8 */     "\x00\xff\xff\x00\xff\xff\xff",
[VR6]   73: /* 9 */     "\x00\x00\x00\xff\x00\xff\xff"
[BD0]   74:       };
[000]   75:
[N36]   76: /* eancode1[][]: Zeichensatz B           */
[MH3]   77: BYTE eancode2[10][7]=
[P70]   78:       {
[4Z6]   79: /* 0 */     "\x00\xff\x00\x00\xff\xff\xff",
[5Z6]   80: /* 1 */     "\x00\xff\xff\x00\x00\xff\xff",
[6Z6]   81: /* 2 */     "\x00\x00\xff\xff\x00\xff\xff",
[3V6]   82: /* 3 */     "\x00\xff\x00\x00\x00\xff\xff",
[8Z6]   83: /* 4 */     "\x00\x00\xff\xff\xff\x00\xff",
[9Z6]   84: /* 5 */     "\x00\xff\xff\xff\x00\x00\xff",
[6V6]   85: /* 6 */     "\x00\x00\x00\x00\xff\xff\xff",
[7V6]   86: /* 7 */     "\x00\x00\xff\x00\x00\x00\xff",
[8V6]   87: /* 8 */     "\x00\x00\x00\xff\x00\x00\xff",
[XT6]   88: /* 9 */     "\x00\x00\xff\x00\xff\xff\xff"
[BD0]   89:       };
[000]   90:
[S36]   91: /* eancode1[][]: Zeichensatz C           */
[PH3]   92: BYTE eancode3[10][7]=
[P70]   93:       {
[4Z6]   94: /* 0 */     "\xff\xff\xff\x00\x00\xff\x00",
[5Z6]   95: /* 1 */     "\xff\xff\x00\x00\xff\xff\x00",
[6Z6]   96: /* 2 */     "\xff\xff\x00\xff\xff\x00\x00",
[3V6]   97: /* 3 */     "\xff\x00\x00\x00\x00\xff\x00",
[8Z6]   98: /* 4 */     "\xff\x00\xff\xff\xff\x00\x00",
[9Z6]   99: /* 5 */     "\xff\x00\x00\xff\xff\xff\x00",
[6V6]  100: /* 6 */     "\xff\x00\xff\x00\x00\x00\x00",
[7V6]  101: /* 7 */     "\xff\x00\x00\x00\xff\x00\x00",
[8V6]  102: /* 8 */     "\xff\x00\x00\xff\x00\x00\x00",
[XT6]  103: /* 9 */     "\xff\xff\xff\x00\xff\x00\x00"
[BD0]  104:       };
[000]  105:
[3Z6]  106: /* initgraf: Drucker->Grafik einstellen  */
[263]  107: BYTE initgraf[10];
[000]  108:
[DT6]  109: /* x,y: Koordinaten für Bildschirm       */
[062]  110: unsigned x,y;
[000]  111:
[8F4]  112: /* DEFAULT-EAN                           */
[UX3]  113: char ean[14]="4902580301620";
[000]  114:
[0N4]  115: main(int argc, char *argv[])
[P70]  116: {
[BP1]  117: int i,z,b;
[HP1]  118: char e[7];
[000]  119:
[871]  120: x=y=20;
[7H2]  121: screen_init();
[000]  122:
[000]  123:
[VF5]  124: sprintf(initgraf,"%c%c%c%c%c",ESC,'*',
[U44]  125:                 M3,SPALTEN%256,SPALTEN/256);
[000]  126:
[R47]  127: if(argv[1]!=NULL) /* ARGUMENT übergeben  */
[YC3]  128:       strcpy(ean,argv[1]);
[XM1]  129: ean[13]=0;
[000]  130:
[XC6]  131: fprintf(stdprn,"%cA%c",ESC,7);/* LF=7/60"*/
[GE2]  132: for(z=0;z<5;z++)
[P70]  133:    {
[ME2]  134:    /* GRAFIK ein    */
[BZ5]  135:    fwrite((BYTE *) initgraf,1,5,stdprn);
[WV2]  136:           /* Zeile drucken */
[VC5]  137:    eanzeile();          /* Zeile drucken */
[NP0]  138:    x=20;
[SX2]  139:    /* MicroLINEFEED */
[B95]  140:    fprintf(stdprn,"%cJ%c\r",ESC,1);
[ME2]  141:    /* GRAFIK ein    */
[BZ5]  142:    fwrite((BYTE *) initgraf,1,5,stdprn);
[VC5]  143:    eanzeile();          /* Zeile drucken */
[R36]  144:    fwrite("\x0a\x0d",1,2,stdprn);/* CRLF */
```

```
[NP0]  145:    y+=8;
[NP0]  146:    x=20;
[U70]  147:    }
[ME2]  148: /* GRAFIK ein    */
[BZ5]  149: fwrite((BYTE *) initgraf,1,5,stdprn);
[000]  150:
[N07]  151: /* Vorrücken bis zum Linken Randzeichen */
[KN2]  152: for(i=0;i<7*4;i++)
[Z26]  153:    fwrite((BYTE *) &ean_null,1,1,stdprn);
[000]  154:
[G96]  155: ean_rand();       /* Linkes Randzeichen */
[000]  156:
[PE6]  157: /* Vorrücken bis zum Trennzeichen       */
[MU2]  158: for(i=0;i<6*7*4;i++)
[Z26]  159:    fwrite((BYTE *) &ean_null,1,1,stdprn);
[000]  160:
[895]  161: trenn();          /*   Trennzeichen     */
[000]  162:
[MZ6]  163: /* Vorrücken bis zum Rechten Randzeichen*/
[MU2]  164: for(i=0;i<6*7*4;i++)
[Z26]  165:    fwrite((BYTE *) &ean_null,1,1,stdprn);
[000]  166:
[2C6]  167: ean_rand();       /* Rechtes Randzeichen */
[000]  168:
[SX2]  169: /* MicroLINEFEED */
[B95]  170: fprintf(stdprn,"%cJ%c\r",ESC,1);
[000]  171:
[ME2]  172: /* GRAFIK ein    */
[BZ5]  173: fwrite((BYTE *) initgraf,1,5,stdprn);
[000]  174:
[N07]  175: /* Vorrücken bis zum Linken Randzeichen */
[KN2]  176: for(i=0;i<7*4;i++)
[Z26]  177:    fwrite((BYTE *) &ean_null,1,1,stdprn);
[000]  178:
[G96]  179: ean_rand();       /* Linkes Randzeichen */
[000]  180:
[PE6]  181: /* Vorrücken bis zum Trennzeichen       */
[MU2]  182: for(i=0;i<6*7*4;i++)
[Z26]  183:    fwrite((BYTE *) &ean_null,1,1,stdprn);
[000]  184:
[895]  185: trenn();          /*   Trennzeichen     */
[000]  186:
[MZ6]  187: /* Vorrücken bis zum Rechten Randzeichen*/
[MU2]  188: for(i=0;i<6*7*4;i++)
[Z26]  189:    fwrite((BYTE *) &ean_null,1,1,stdprn);
[000]  190:
[2C6]  191: ean_rand();       /* Rechtes Randzeichen */
[000]  192:
[000]  193:
[8G6]  194: /* für Ausdruck in Arabischen Ziffern   */
[RN4]  195: /* EAN aufbereiten                      */
[DX2]  196: strcpy(e,&ean[1]);
[661]  197: e[6]=0;
[XM1]  198: ean[13]=0;
[000]  199:
[SX2]  200: /* MicroLINEFEED */
[H95]  201: fprintf(stdprn,"%cJ%c\r",ESC,3);
[000]  202:
[7Z6]  203: /*EAN in ARABISCHEN Ziffern FETT drucken*/
[2D5]  204: fprintf(stdprn,"%cE%c  %6s   %6s%cF",
[564]  205:      ESC,ean[0],e,&ean[7],ESC);
[000]  206:
[HT4]  207: fwrite("\x0a\x0d",1,2,stdprn);
[000]  208:
[7M2]  209: /* LINEFEED=1/6" */
[TR2]  210: /* = standard LF */
[X74]  211: fprintf(stdprn,"%c2",ESC);
[000]  212:
[MP6]  213: getchar(); /* Warte auf <RETURN>-Taste  */
[RH2]  214: screen_exit();
[U70]  215: }
[000]  216:
[PB2]  217: screen_init()
[P70]  218: {
[XN3]  219: struct videoconfig vc;
[FT1]  220: int modus;
[UT5]  221:     /* suche BESTEN Video GRAFIK-Modus */
[LX6]  222: for(modus=_VRES16COLOR;modus>=0;modus--)
[495]  223:      if(_setvideomode(modus)) break;
[000]  224:
[CL3]  225: _getvideoconfig(&vc);
[000]  226:
[1U4]  227: /* Bildschirm weiß füllen */
[U94]  228: _setcolor(_BRIGHTWHITE);
[N05]  229: _rectangle(_GFILLINTERIOR,0,0,
[T65]  230:         vc.numxpixels,vc.numypixels);
[U70]  231: }
[000]  232:
[6C2]  233: screen_exit()
[P70]  234: {
[ML6]  235: /* Bildschirm in TEXT-Mode rücksetzten */
[4Y4]  236: _setvideomode(_DEFAULTMODE);
[U70]  237: }
[000]  238:
[KM1]  239: ean_rand()
[P70]  240: {
[111]  241: int b;
[000]  242:
[S92]  243: for(b=0;b<3;b++)
[P70]  244:    {
```

```
[X26]  245: /* jedes Modul des Randzeichens 4 mal  */
[E66]  246: /* ausgeben, um lesbare Balken zu er-  */
[VY3]  247: /* reichen.                           */
[SD6]  248:    fwrite((BYTE *) &eanrand[b],1,1,stdprn);
[SD6]  249:    fwrite((BYTE *) &eanrand[b],1,1,stdprn);
[SD6]  250:    fwrite((BYTE *) &eanrand[b],1,1,stdprn);
[SD6]  251:    fwrite((BYTE *) &eanrand[b],1,1,stdprn);
[000]  252:
[KY5]  253:    /* Keine Randzeichen am Bildschirm  */
[4R5]  254:    /* wenn 8 Zeilen augegeben wurden   */
[893]  255:    if(y-20>=40) continue;
[000]  256:
[MN6]  257:  /* Randzeichen am Bildschirm darstellen */
[CD3]  258:    if(eanrand[b]==0xff)
[053]  259:       _setcolor(_BLACK);
[ZR0]  260:    else
[U94]  261:       _setcolor(_BRIGHTWHITE);
[WW5]  262:    _moveto(x,y); _lineto(x,y+8); x+=1;
[WW5]  263:    _moveto(x,y); _lineto(x,y+8); x+=1;
[SX4]  264:    _moveto(x,y); _lineto(x,y+8);
[U70]  265:    }
[U70]  266: }
[000]  267:
[S51]  268: trenn()
[P70]  269: {
[111]  270: int b;
[0A2]  271: for(b=0;b<5;b++)
[P70]  272:    {
[DH6]  273:    fwrite((BYTE*) &eantrenn[b],1,1,stdprn);
[DH6]  274:    fwrite((BYTE*) &eantrenn[b],1,1,stdprn);
[DH6]  275:    fwrite((BYTE*) &eantrenn[b],1,1,stdprn);
[DH6]  276:    fwrite((BYTE*) &eantrenn[b],1,1,stdprn);
[893]  277:    if(y-20>=40) continue;
[2K3]  278:    if(eantrenn[b]==0xff)
[053]  279:       _setcolor(_BLACK);
[ZR0]  280:    else
[U94]  281:       _setcolor(_BRIGHTWHITE);
[WW5]  282:    _moveto(x,y); _lineto(x,y+8); x+=1;
[WW5]  283:    _moveto(x,y); _lineto(x,y+8); x+=1;
[SX4]  284:    _moveto(x,y); _lineto(x,y+8);
[U70]  285:    }
[U70]  286: }
[000]  287:
[FM6]  288: /* Augabe EAN-Ziffer aus  Zeichensatz A  */
[LL2]  289: eanzifferl(int i)
[P70]  290: {
[111]  291: int b;
[8A2]  292: for(b=0;b<7;b++)
[P70]  293:    {
[SG6]  294:    fwrite((BYTE *) &eancodel[ean[i]-'0'][b]
[Y02]  295:                   ,1,1,stdprn);
[SG6]  296:    fwrite((BYTE *) &eancodel[ean[i]-'0'][b]
[Y02]  297:                   ,1,1,stdprn);
[SG6]  298:    fwrite((BYTE *) &eancodel[ean[i]-'0'][b]
[Y02]  299:                   ,1,1,stdprn);
[SG6]  300:    fwrite((BYTE *) &eancodel[ean[i]-'0'][b]
[Y02]  301:                   ,1,1,stdprn);
[893]  302:    if(y-20>=40) continue;
[KB5]  303:    if(eancodel[ean[i]-'0'][b]==0xff)
[053]  304:       _setcolor(_BLACK);
[ZR0]  305:    else
[U94]  306:       _setcolor(_BRIGHTWHITE);
[WW5]  307:    _moveto(x,y); _lineto(x,y+8); x+=1;
[WW5]  308:    _moveto(x,y); _lineto(x,y+8); x+=1;
[SX4]  309:    _moveto(x,y); _lineto(x,y+8);
[U70]  310:    }
[U70]  311: }
[000]  312:
[LM6]  313: /* Augabe EAN-Ziffer aus  Zeichensatz B  */
[PL2]  314: eanziffer2(int i)
[P70]  315: {
[111]  316: int b;
[8A2]  317: for(b=0;b<7;b++)
[P70]  318:    {
[UG6]  319:    fwrite((BYTE *) &eancode2[ean[i]-'0'][b]
[Y02]  320:                   ,1,1,stdprn);
[UG6]  321:    fwrite((BYTE *) &eancode2[ean[i]-'0'][b]
[Y02]  322:                   ,1,1,stdprn);
[UG6]  323:    fwrite((BYTE *) &eancode2[ean[i]-'0'][b]
[Y02]  324:                   ,1,1,stdprn);
[UG6]  325:    fwrite((BYTE *) &eancode2[ean[i]-'0'][b]
[Y02]  326:                   ,1,1,stdprn);
[893]  327:    if(y-20>=40) continue;
[PB5]  328:    if(eancode2[ean[i]-'0'][b]==0xff)
[053]  329:       _setcolor(_BLACK);
[ZR0]  330:    else
[U94]  331:       _setcolor(_BRIGHTWHITE);
[WW5]  332:    _moveto(x,y); _lineto(x,y+8); x+=1;
[WW5]  333:    _moveto(x,y); _lineto(x,y+8); x+=1;
[SX4]  334:    _moveto(x,y); _lineto(x,y+8);
[U70]  335:    }
[U70]  336: }
[000]  337:
[RM6]  338: /* Augabe EAN-Ziffer aus  Zeichensatz C  */
[TL2]  339: eanziffer3(int i)
[P70]  340: {
[111]  341: int b;
[8A2]  342: for(b=0;b<7;b++)
[P70]  343:    {
[9E6]  344:    fwrite((BYTE*)&eancode3[ean[i]-'0'][b],
[5U1]  345:                   1,1,stdprn);
[9E6]  346:    fwrite((BYTE*)&eancode3[ean[i]-'0'][b],
[5U1]  347:                   1,1,stdprn);
[9E6]  348:    fwrite((BYTE*)&eancode3[ean[i]-'0'][b],
[5U1]  349:                   1,1,stdprn);
[9E6]  350:    fwrite((BYTE*)&eancode3[ean[i]-'0'][b],
[5U1]  351:                   1,1,stdprn);
[893]  352:    if(y-20>=40) continue;
[UB5]  353:    if(eancode3[ean[i]-'0'][b]==0xff)
[053]  354:       _setcolor(_BLACK);
[ZR0]  355:    else
[U94]  356:       _setcolor(_BRIGHTWHITE);
[WW5]  357:    _moveto(x,y); _lineto(x,y+8); x+=1;
[WW5]  358:    _moveto(x,y); _lineto(x,y+8); x+=1;
[SX4]  359:    _moveto(x,y); _lineto(x,y+8);
[U70]  360:    }
[U70]  361: }
[000]  362:
[0T6]  363: /* Druckausgabe einer ganzen EAN-Zeile   */
[4M1]  364: eanzeile()
[P70]  365: {
[F11]  366: int i;
[HV6]  367: /* Vorrücken zum Linken Randzeichen      */
[KN2]  368: for(i=0;i<7*4;i++)
[Z26]  369:    fwrite((BYTE *) &ean_null,1,1,stdprn);
[000]  370:
[E05]  371: /* Linkes Randzeichen drucken */
[YV1]  372: ean_rand();
[000]  373:
[PB2]  374: for(i=1;i<7;i++)
[P70]  375:    {
[6A5]  376:    /* je nach 13.Stelle (==ean[0])       */
[HA5]  377:    if(eanparity[ean[0]-'0'][i-1]==1)
[G36]  378:       /*Zeichen aus Zeichensatz A drucken*/
[362]  379:       eanzifferl(i);
[ZR0]  380:    else
[M36]  381:       /*Zeichen aus Zeichensatz B drucken*/
[662]  382:       eanziffer2(i);
[U70]  383:    }
[000]  384:
[X74]  385: /* Trennzeichen drucken */
[L71]  386: trenn();
[000]  387:
[0G2]  388: for(i=7;i<13;i++)
[S36]  389:    /*Zeichen aus Zeichensatz C drucken*/
[962]  390:    eanziffer3(i);
[000]  391:
[705]  392: /* Rechtes Randzeichen drucken */
[YV1]  393: ean_rand();
[U70]  394: }
```

Listing 1. »eandemo.c« demonstriert die EAN-Code-Ausgabe

```
[6B3]    1: ; Programm: scan.asm
[197]    2: ; Funktion: stellt eine Interrupt-Service-
[4G6]    3: ; Routine für READEAN.C zur Verfügung
[HD6]    4: ; An der Schnittstelle COM1 muß ein
[MP6]    5: ; EAN-Lesegerät zur Verfügung stehen.
[M06]    6: ; COM1 wird von READEAN.C mit 9600 baud
[NE5]    7: ; 8 Datenbits, NOPARITY, 1 STOPBIT
[VD6]    8: ; initialisiert. Soll das Programm mit
[F07]    9: ; COM2 arbeiten, müssen die Portadressen
[456]   10: ; entsprechend dem Kommentar geändert
[5D1]   11: ; werden.
[AF5]   12: ; Assemblieren mit: masm /mx scan;
[605]   13: ; Linken mit: link readean scan;
[1F2]   14: ; Sprache: Masm
[L53]   15: ; Autor: Josef Eder
[FU4]   16: ; (c)1991 DMV Widuch GmbH & Co. KG
[P30]   17: ;
[T45]   18: _TEXT SEGMENT BYTE PUBLIC 'CODE'
[EU2]   19:       ASSUME CS:_TEXT
[FU2]   20:       ASSUME DS:_TEXT
[3T5]   21:       PUBLIC _asm_init,_com_interrupt
[000]   22:
[1S2]   23: eoi             equ 20h
[155]   24: ctrl            equ 20h; Adresse 8259a
[1K4]   25: rbr             dw 3f8h; COM2: 2f8h
[T64]   26: iir             dw 3fah;       2fah
[N84]   27: lsr             dw 3fdh;       2fdh
[000]   28:
[9L2]   29: buffer          dw 0
[NK2]   30: bufseg          dw 0
[5N2]   31: pointr          dw 0
[GK2]   32: bufend          dw 0
[HM4]   33: ;---------------------------------------
[CB6]   34: ; Die Funktion ASM_INIT wird von C aus
[2Y5]   35: ; aufgerufen. Die Parameter : eanptr,
[EX5]   36: ; eanbuf werden am STACK uebergeben.
[8G2]   37: ; AUFRUF VON C:
[5M4]   38: ;     asm_init(eanptr,eanbuf);
[HM4]   39: ;---------------------------------------
[AW1]   40: _asm_init:
[TB1]   41:     push bp
```

Listing 2. »scan.asm« stellt für »readean.c« eine Interrupt-Serviceroutine zur Verfügung, um EAN-Codes einlesen zu können

```
[PG1]   42:     mov bp,sp
[ØF6]   43: ; bp=sp setzen, in bp+4 steht die Adresse
[EB2]   44: ; des Pointers
[252]   45:     mov ax,[bp+4]
[ZT2]   46:     mov cs:pointr,ax
[CN6]   47: ; in bp+6 steht die Adresse des Buffers
[YP5]   48: ; in den der EAN-CODE gelesen werden
[DZØ]   49: ; soll
[452]   5Ø:     mov ax,[bp+6]
[5S2]   51:     mov cs:buffer,ax
[CC4]   52: ; Ende des EANBUFFERS ist bei
[B65]   53: ; buffer+15 (13 Stellen + 1 CR + 1 LF)
[LA1]   54:     add ax,15
[ER2]   55:     mov cs:bufend,ax
[TH5]   56: ; Datensegment des Buffers sichern
[3H1]   57:     mov ax,ds
[3T2]   58:     mov cs:bufseg,ax
[131]   59:     pop bp
[3PØ]   6Ø:     ret
[AL4]   61: ; ----------------------------------------
[XB7]   62: ; com_interrupt wird von aus C installiert
[4R2]   63: ; ACHTUNG: ds = cs
[AL4]   64: ; ----------------------------------------
[PY2]   65: _com_interrupt:
[TB1]   66:     push bp
[BC1]   67:     push ds
[PC1]   68:     push ax
[TC1]   69:     push bx
[ZC1]   7Ø:     push dx
[EC1]   71:     push es
[3B1]   72:     push di
[ØØØ]   73:
[XG6]   74: ; Datum aus Interrupt-Identification-
[SM2]   75: ; Register holen
[772]   76:     mov dx,cs:IIR
[161]   77:     in al,dx
[9R4]   78: ; wenn Bit 2 nicht gesetzt ist
[BF6]   79: ; (dh. Empfangsregister noch nicht voll)
[Ø74]   8Ø: ; dann springe zu end_intr
[LK1]   81:     cmp al,Ø4h
[782]   82:     jnz end_intr
[RR6]   83: ; Datum aus Line-Status-Register holen
[X72]   84:     mov dx,cs:LSR
[161]   85:     in al,dx
[7R4]   86: ; wenn Bit Ø nicht gesetzt ist
[994]   87: ; (dh. kein Datum empfangen)
[Ø74]   88: ; dann springe zu end_intr
[WG1]   89:     and al,Ø1h
[M32]   9Ø:     jz end_intr
[L45]   91: ; es muß ds werden (siehe oben)
[4S2]   92:     mov ds,cs:bufseg
[BC1]   93:     push ds
[G31]   94:     pop es
[456]   95: ; Datum aus Receive-Buffer-Register
[4B6]   96: ; holen - hier steht jetzt die EAN-Ziffer
[482]   97:     mov dx,cs:RBR
[161]   98:     in al,dx
[ZV2]   99:     mov bx,cs:pointr
[X46]  1ØØ: ; Adresse ES:DI ist der nächst freie
[ØR4]  1Ø1: ; Platz für die EAN-Ziffer
[D32]  1Ø2:     mov di,[bx]
[1M4]  1Ø3: ; EAN-Ziffer in ES:DI ablegen
[NXØ]  1Ø4:     stosb
[W65]  1Ø5: ; di wurde bei stosb erhoeht dh.:
[3Ø7]  1Ø6: ; ES:DI ist Adresse des nächsten freien
[AL5]  1Ø7: ; Platzes -> DI (offset) speichern
[Ø42]  1Ø8:     mov [bx],di
[U76]  1Ø9: ; in BX steht der Offset der aktuellen
[765]  11Ø: ; Adresse. Diese wird mit buffend
[512]  111: ; verglichen
[6S2]  112:     mov bx,cs:bufend
[ZE1]  113:     sub bx,di
[5L6]  114: ; wenn kleiner oder gleich dann end_intr
[B52]  115:     jnc end_intr
[7N6]  116: ; ansonsten pointr auf Adresse von buffer
[2T6]  117: ; setzen, sichern und ebenfalls end_intr
[PU2]  118:     mov di,cs:pointr
[HR2]  119:     mov ax,cs:buffer
[ZYØ]  12Ø:     stosw
[NL1]  121: end_intr:
[G22]  122:     mov dx,ctrl
[AG4]  123: ; Zum Interruptkontroller
[V34]  124: ; End-OF-Interrupt senden
[LP1]  125:     mov al,eoi
[6H1]  126:     out dx,al
[G21]  127:     pop di
[G31]  128:     pop es
[X31]  129:     pop dx
[T31]  13Ø:     pop bx
[R31]  131:     pop ax
[E31]  132:     pop ds
[131]  133:     pop bp
[815]  134: ; Interrupts wieder zulassen
[6PØ]  135:     sti
[GRØ]  136:     iret
[7X1]  137: _TEXT  ENDS
[4MØ]  138:        END
```

Listing 2. »scan.asm« stellt für »readean.c« eine Interrupt-Serviceroutine zur Verfügung, um EAN-Codes einlesen zu können

```
[VE3]    1: /* Programm: readean.c
[CP6]    2:    Funktion:  Demonstriert die Verwendung
[CH4]    3:    eines EAN-Code-Lesegeräts.
[H96]    4:    Nach dem Start der erzeugten EXE-Datei
[W46]    5:    wartet das Programm auf das Einlesen
[FR5]    6:    eines EAN-Codes durch ein EAN-Lese-
[5R5]    7:    gerät. Danach wird der EAN-Code aus-
[GE6]    8:    gegeben und das Programm wird beendet.
[T56]    9:    Das Programm kann durch Tastendruck
[G34]   1Ø:    ebenfalls beendet werden.
[445]   11:    Compilieren mit: qcl readean.c
[TS4]   12:    Linken mit: link readean scan
[5P2]   13:    Sprache: Quick C
[NT2]   14:    Autor: Josef Eder
[T15]   15:    (c)1991 DMV Widuch GmbH & Co. KG */
[ØØØ]   16:
[L13]   17: #include <stdio.h>
[9L2]   18: #include <dos.h>
[AØ3]   19: #include <ctype.h>
[2Ø3]   2Ø: #include <conio.h>
[ØØØ]   21:
[XD6]   22: /* Portadresser der 1. Seriellen (COM1) */
[SP2]   23: #define RBR Øx3f8
[HP2]   24: #define IER Øx3f9
[DR2]   25: #define IIR Øx3fa
[3R2]   26: #define LCR Øx3fb
[7R2]   27: #define MCR Øx3fc
[RS2]   28: #define LSR Øx3fd
[VS2]   29: #define MSR Øx3fe
[ØØØ]   3Ø:
[ØT5]   31: /* eine Zeitverzögerung                  */
[656]   32: #define DELAY for(dly=Ø;dly<32ØØØ;dly++)
[BX2]   33: unsigned int dly;
[ØØØ]   34:
[MF5]   35: /* An asm_init() (SCAN.ASM) wird der
[5D5]   36:    Puffer für 15 EANs und ein Pointer
[HA5]   37:    auf diesen Puffer übergeben   */
[VS3]   38: extern int asm_init();
[ØØØ]   39:
[YE6]   4Ø: /* com_interrupt (SCAN.ASM) wird aufge-
[9Ø5]   41:    rufen wenn an COM1 ein Hardware-
[TN3]   42:    interrupt eintrifft.
[M56]   43:    Trifft der Interrupt ein, so schreibt
[LG6]   44:    com_interrupt() die ankommenden BYTES
[ER5]   45:    auf den bei asm_init() übergebenen
[9K6]   46:    Puffer; der Pointer der bei asm_init()
[GU5]   47:    übergeben wurde wird erhöht, und
[N16]   48:    signalisiert damit der routine scan()
[FZ5]   49:    daß nun ein 13-stelliger EAN-Code
[ØN4]   5Ø:    in eanbuf[] bereitsteht.  */
[KC7]   51: extern void interrupt far com_interrupt();
[ØØØ]   52:
[8Y6]   53: /* Interrupts werden durch Programmierung
[N66]   54:    des 8259a gesperrt bzw. freigegeben.  */
[NF2]   55: enable_ints();
[ØØØ]   56:
[CF6]   57: /* scan(): hat sich eanptr geändert, dann
[717]   58:    ist com_interrupt ausgeführt worden   */
[BF3]   59: unsigned int scann();
[ØØØ]   6Ø:
[FW6]   61: /* isean(): prüft ob ein korrekter EANcode
[RØ7]   62:    gelesen wurde (Scanning hat funktioniert
[RL6]   63:    Sollte scan() zu schnell sein so stehen
[MS6]   64:    in manchen BYTES von eanbuf[] wahrschein
[W46]   65:    lich Binäre NULLEN; es sind aber nur
[8K3]   66:    ASCII-NULLEN erlaubt:
[CP5]   67:    Bei Fehler wird Ø (für NICHT EAN) bei
[U44]   68:    Erfolg 1 zurückgegeben */
[TU2]   69: int isean(char *);
[ØØØ]   7Ø:
[LL5]   71: /* scanner_set(): bereitet COM1 auf
[SØ4]   72:    Empfang von Daten vor.*/
[3E2]   73: scanner_set();
[ØØØ]   74:
[PH6]   75: /* scan_prep(): initialisier Puffer und
[3V1]   76:    Pointer.  */
[LZ1]   77: scan_prep();
[ØØØ]   78:
[1H6]   79: /* int_restore(): setzt den Interrupt-
[VM6]   8Ø:    vektor auf die ursprüngliche Interrupt
[Y43]   81:    routine zurück. */
[WC3]   82: void int_restore();
[ØØØ]   83:
[LS5]   84: /* int_inst(): installiert die neue
[H96]   85:    Interruptroutine com_interrupt() */
[BP2]   86: void int_inst();
[ØØØ]   87:
[AT5]   88: /* hier merken wir uns die Adresse der
[MS6]   89:    ursprünglichen Interruptroutine.      */
[6B5]   9Ø: void (interrupt far *alt_int)();
[ØØØ]   91:
[AM5]   92: /* Auf den beiden folgenden Adressen
[2Y5]   93:    stehen Pointer auf Begin und Ende des
[Z26]   94:    Tastaturpuffers. Sobald eine Taste
[AC6]   95:    gedrückt wird ändert sich der Inhalt
[FF5]   96:    von *tast_ptr2 dh. *tast_ptrl ist
[6D6]   97:    ungleich *tast_ptr2: "EINE TASTE WURDE
[Z34]   98:    GEDRÜCKT !!!"                         */
[7T4]   99: unsigned int far *tast_ptrl
[PS4]  1ØØ:      =(unsigned int far *) Øx4ØØØ1aL;
```

```
[BT4]  101: unsigned int far *tast_ptr2
[SS4]  102:      =(unsigned int far *) 0x00400001cL;
[000]  103:
[XP6]  104: /* in eanbuf[] steht der EAN-Code nachdem*/
[G47]  105: /* COM1 einen Interrupt erzeugt hat und  */
[FS6]  106: /* com_interrupt 15 BYTES von COM1 abge- */
[1D4]  107: /* holt hat.                             */
[854]  108: unsigned char eanbuf[15];
[9B5]  109: /* Zeiger auf eanbuf[];                  */
[DL5]  110: /* wird von scan() abgefragt             */
[6P5]  111: /*    if(eanptr!=&eanbuf[0]) ...         */
[9N3]  112: unsigned char *eanptr;
[000]  113:
[KZ6]  114: /* in status wird das StatusRegister des */
[GV6]  115: /* Interruptcontrollers 8259a gemerkt.   */
[N32]  116: int status;
[000]  117:
[UW0]  118: main()
[P70]  119: {
[BH1]  120: int ende;
[000]  121:
[3E2]  122: scanner_set();
[YZ1]  123: int_inst();
[Z51]  124: ende=0;
[000]  125:
[LE6]  126: printf("Bitte lesen Sie jetzt mit\n");
[5N6]  127: printf("Ihrem EAN-Lesegerät einen\n");
[SD4]  128: printf("EAN-Code ein ...\n");
[WK6]  129: printf("oder drücken Sie eine Taste\n");
[LA6]  130: printf("um das Programm zu beenden.\n");
[373]  131: printf("\n\n\n->");
[571]  132: while(1)
[P70]  133:    {
[LZ1]  134:    scan_prep();
[4G3]  135:    while(!(ende=scann())) ;
[L53]  136:    if(ende==0xff) break;
[NF2]  137:    enable_ints();
[PC5]  138:    printf("\nEAN:%15s\n",&eanbuf[0]);
[RT2]  139:    if(!isean(eanbuf))
[P70]  140:       {
[5R5]  141:       printf("Fehler beim Einlesen !!!\n");
[SE5]  142:       printf("Bitte WIEDERHOLEN !!!\n");
[AM1]  143:       continue;
[U70]  144:       }
[ZR0]  145:    else
[P01]  146:       break;
[U70]  147:    }
[HG2]  148: int_restore();
[U70]  149: }
[000]  150:
[K82]  151: scanner_set()
[P70]  152: {
[B55]  153: /* COM1 vorbereiten:                      */
[F56]  154: /* 9600 Baud 8 Bit NoParity 1 Stopbit    */
[NV1]  155: _disable();
[PG2]  156: outp(LCR,0x80);
[NW1]  157: outp(IER,0);
[2K2]  158: outp(RBR,0x0c);
[ZW1]  159: outp(LCR,3);
[NW1]  160: outp(IER,0);
[DN1]  161: _enable();
[U70]  162: }
[000]  163:
[GM2]  164: void int_inst()
[P70]  165: {
[NV1]  166: _disable();
[000]  167:
[MV5]  168: /* alten Interruptvektor merken  */
[AV4]  169: alt_int=_dos_getvect(0x0c);
[000]  170:
[ES5]  171: /* neuen Interrupt installieren  */
[BX5]  172: _dos_setvect(0x0c,com_interrupt);
[000]  173:
[A05]  174: /* Datenport des 8259a merken    */
[ZP2]  175: status=inp(0x21);
[000]  176:
[1B6]  177: /* 8250 auf Hardware Interrupts vor- */
[K24]  178: /* bereiten.                         */
[NH2]  179: outp(MCR,0x0B);
[000]  180:
[DN1]  181: _enable();
[U70]  182: }
[000]  183:
[G53]  184: void int_restore()
[P70]  185: {
[NV1]  186: _disable();
[000]  187:
[EX6]  188: /* Alten Status um Interrupt-Controller-
[A95]  189:    Datenport wieder herstellen. */
[X23]  190: outp(0x21,status);
[000]  191:
[XS6]  192: /* Hardwareninterrupts am 8250 verbieten */
[NW1]  193: outp(IER,0);
[RG2]  194: outp(MCR,0x03);
[000]  195:
[FG6]  196: /* Alte Interruptroutine aktivieren */
[3T4]  197: _dos_setvect(0x0c,alt_int);
[000]  198:
[DN1]  199: _enable();
[U70]  200: }
```

```
[000]  201:
[SX1]  202: scan_prep()
[P70]  203: {
[VV3]  204: /* EANpuffer löschen */
[003]  205: memset(eanbuf,0,15);
[000]  206:
[HY6]  207: /* eanptr zeigt auf Beginn des Eanpuffers*/
[R63]  208: eanptr=&eanbuf[0];
[000]  209:
[1R6]  210: /* übergebe Puffer und zeiger darauf an  */
[ZC4]  211: /* die ASSEMBLER-Routinen. */
[3A4]  212: asm_init(&eanptr,eanptr);
[000]  213:
[FN5]  214: /* sicherheitshalber ein BYTE vom */
[CS6]  215: /* RecieveBufferRegister des 8250 abholen*/
[NH1]  216: inp(RBR);
[000]  217:
[NV6]  218: /* erlaube nur Tastaturinterrupt UND ... */
[ZE2]  219: outp(0x21,0xec);
[000]  220:
[EN4]  221: /* erlaube Interrupt am 8250 */
[SW1]  222: outp(IER,1);
[U70]  223: }
[000]  224:
[4A2]  225: enable_ints()
[P70]  226: {
[8F6]  227: /* Alter Status ins Datenport des 8259 */
[X23]  228: outp(0x21,status);
[000]  229:
[WZ4]  230: /* verbiete Interrupts am 8250 */
[NW1]  231: outp(IER,0);
[U70]  232: }
[000]  233:
[NB3]  234: unsigned int scann()
[P70]  235: {
[ZS6]  236: /* Wenn Inhalt der beiden Pointer ungleich
[PM6]  237:    dann ist eine Taste gedrückt worden  */
[SK4]  238: if(*tast_ptr1 != *tast_ptr2)
[P70]  239:    {
[NF2]  240:    enable_ints();
[000]  241:
[EA6]  242:    /* Zeichen aus Tastaturpuffer abholen */
[R61]  243:    getch();
[000]  244:
[8R5]  245:    /* gib Programmende FLAG zurück    */
[F52]  246:    return(0xff);
[U70]  247:    }
[000]  248:
[EH6]  249: /* Zeigt enaptr NICHT auf Anfangsadresse
[346]  250:    von eanbuf, dann hat ein Interrupt an
[Z76]  251:    COM1 com_interrupt() aufgerufen und
[UU5]  252:    ccm_interrupt hat Zeichen von COM1
[G17]  253:    abgeholt und in eanbuf[] bereitgestellt
[170]  254: */
[YL3]  255: if(eanptr!=&eanbuf[0])
[P70]  256:    {
[XN5]  257:    /* Zeitlassen um 15 EANs zu lesen  */
[B11]  258:    DELAY;
[SS5]  259:    /* gib 1 für ean gelesen zurück    */
[UL1]  260:    return(1);
[U70]  261:    }
[ZR0]  262: else
[8P6]  263:  /* weder Taste gedrückt noch ean gelesen*/
[TL1]  264:    return(0);
[U70]  265: }
[000]  266:
[483]  267: int isean(char *ean)
[P70]  268: {
[F11]  269: int i;
[4F2]  270: for(i=0;i<13;i++)
[2D4]  271:    if(*(ean+i)<'0' || *(ean+i)>'9')
[TL1]  272:       return(0);
[UL1]  273: return(1);
[U70]  274: }
```

Listing 3. Die Verarbeitung eingelesener EAN-Codes übernimmt »readean.c«

```
[6G5]    1: 80 0A 00 08 73 63 61 6E 2E 61 73 6D 5A 96 0D 00
[DC5]    2: 00 04 43 4F 44 45 05 5F 54 45 58 54 95 98 07 00
[3C5]    3: 28 82 00 03 02 01 B1 90 22 00 00 01 0E 5F 63 6F
[LK5]    4: 6D 5F 69 6E 74 65 72 72 75 70 74 2E 00 00 09 5F
[UF5]    5: 61 73 6D 5F 69 6E 69 74 0E 00 00 5D 88 04 00 00
[AD5]    6: A2 01 D1 A0 86 00 01 00 00 F8 03 FA 03 FD 03 00
[TA5]    7: 00 00 00 00 00 00 00 55 8B EC 8B 46 04 2E A3 00
[SD5]    8: 00 8B 46 06 2E A3 00 00 05 0F 00 2E A3 00 00 8C
[8K5]    9: D8 2E A3 00 00 5D C3 55 1E 50 53 52 06 57 2E 8B
[4E5]   10: 16 00 00 EC 3C 04 75 34 2E 8B 16 00 00 EC 24 01
[EM5]   11: 74 2A 2E 8E 1E 00 00 1E 07 2E 8B 16 00 00 EC 2E
[2S5]   12: 8B 1E 00 00 8B 3F AA 89 3F 2E 8B 1E 00 00 2B DF
[4L5]   13: 73 0A 2E 8B 3E 00 00 2E A1 00 00 AB BA 20 00 B0
[1T5]   14: 20 EE 5F 07 5A 5B 58 1F 5D FB CF B9 9C 53 00 C4
[H85]   15: 16 00 01 01 0A 00 C4 1D 00 01 01 06 00 C4 24 00
[H65]   16: 01 01 0C 00 C4 2A 00 01 01 08 00 C4 38 00 01 01
[855]   17: 02 00 C4 42 00 01 01 04 00 C4 4C 00 01 01 08 00
[S55]   18: C4 53 04 01 01 C4 59 00 01 01 0A 00 C4 63 00 01
[FA5]   19: 01 0C 00 C4 6C 00 01 01 0A 00 C4 70 00 01 01 06
[G62]   20: 00 3B 8A 02 00 00 74
```

Listing 4. »scan.obj« ist die linkbare Datei für »readean.c«



MASM ab Version 4.0

Keine Angst vor großen Bäumen

Wer kennt sie nicht: die großen Verzeichnisbäume für Fonts, Treiber, Bilder in verschiedenen Auflösungen und so weiter? Nachdem man sie beispielsweise mit »xcopy« kopiert hat, möchte man den Datenträger von ihnen befreien. An diesem Punkt beginnt dann ein sehr mühsames Löschen von Dateien (eventuell erst nach Attributsänderung) und Verzeichnissen. »direx« zeigt, wie es sehr viel einfacher funktioniert.

Das Löschen von Verzeichnisbäumen ist meistens eine nervtötende Angelegenheit. Das Programm »direx.com« (Listing 1) übernimmt diese Aufgabe. Der Aufruf lautet:

```
direx [Laufwerk]:\[pfad]
```

»\« und Stellvertreterzeichen (Wildcards) (»*.*«) fügt das Programm dem angegebenen Verzeichnis an. Aus Sicherheitsgründen bricht das Programm bei Eingabe von weniger als zehn Zeichen ab und gibt den korrekten Aufruf aus: Eine Eingabe wie beispielsweise »direx c:\«, bei der die gesamte Festplatte gelöscht würde, ist daher nicht möglich. Das Programm arbeitet in zwei Durchläufen. Ist das angegebene Verzeichnis vorhanden, werden nach dem ersten Durchlauf das Verzeichnis und seine Unterverzeichnisse sowie die Anzahl der Dateien (auch versteckte) angezeigt. Mit [Esc] kann Direx jetzt beendet werden, [Enter] startet den zweiten Durchlauf. Das Verzeichnis, seine Unterverzeichnisse und die darin enthaltenen Dateien, auch »hidden« und »read-only«, werden gelöscht.

Das Assembler-Programm »direx.asm« (Listing 2) kann wie folgt kompiliert und dann in eine Com-Datei umgewandelt werden:

```
tasm direx.asm
tlink direx.obj /t
```

sofern TASM verwendet wird. Der Parameter »/t« erzeugt dann direkt eine Com-Datei – oder mit

```
masm direx.asm;
link direx.obj;
exe2bin direx.exe direx.com
```

(Kurt Müller/bj)

```
[X25]    1: E9 5A ØA ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]    2: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]    3: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]    4: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]    5: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]    6: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]    7: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]    8: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]    9: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   1Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   11: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   12: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   13: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   14: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   15: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   16: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   17: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   18: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   19: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   2Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   21: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   22: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   23: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   24: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   25: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   26: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   27: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   28: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   29: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   3Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   31: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   32: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   33: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   34: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   35: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   36: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   37: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   38: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   39: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   4Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   41: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   42: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   43: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   44: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   45: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   46: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   47: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   48: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   49: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   5Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   51: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   52: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   53: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   54: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   55: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   56: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   57: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   58: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   59: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   6Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   61: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØU5]   62: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   63: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   64: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   65: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   66: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   67: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   68: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   69: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   7Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   71: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   72: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   73: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   74: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   75: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   76: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   77: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   78: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   79: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØM5]   8Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   81: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   82: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   83: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   84: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   85: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   86: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØYK5]  87: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   88: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   89: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   9Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   91: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   92: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   93: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   94: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   95: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   96: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   97: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   98: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   99: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  1ØØ: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  1Ø1: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  1Ø2: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  1Ø3: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  1Ø4: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  1Ø5: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  1Ø6: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  1Ø7: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
```

```
[ØZ4]  1Ø8: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  1Ø9: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  11Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  111: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  112: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  113: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  114: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  115: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  116: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  117: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  118: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  119: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  12Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  121: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  122: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  123: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  124: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  125: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  126: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  127: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  128: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  129: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  13Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  131: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  132: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  133: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  134: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  135: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  136: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  137: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  138: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  139: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  14Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]  141: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[V85]  142: ØØ ØØ ØØ ØØ ØØ ØØ ØØ 41 75 66 72 75 66 3A 2Ø 64 69
[EH5]  143: 72 65 78 2Ø 5B 4C 61 75 66 77 65 72 6B 5D 3A 5C
[1H5]  144: 5B 64 69 72 5D 2Ø ØA ØD FF ØA ØD 2Ø 44 61 74 65
[EC5]  145: 69 2Ø 6F 64 65 72 2Ø 5Ø 66 61 64 2Ø 6E 69 63 68
[NH5]  146: 74 2Ø 67 65 66 75 6E 64 65 6E 2Ø ØA ØD FF ØA ØD
[TD5]  147: 2Ø 56 65 72 7A 65 69 63 68 6E 69 73 2Ø 75 6E 64
[ZF5]  148: 2Ø 61 6C 6C 65 2Ø 55 6E 74 65 72 76 65 72 7A 65
[TH5]  149: 69 63 68 6E 69 73 73 65 2Ø 6C 94 73 63 68 65 6E
[WD5]  15Ø: 2Ø 3F 2Ø 3F 2Ø 3F 2Ø ØA ØD 2Ø 41 62 62 72 65 63
[8C5]  151: 68 65 6E 2Ø 3D 2Ø 5B 45 73 63 5D 2Ø 2Ø 2Ø 2Ø 4C
[LK5]  152: 94 73 63 68 65 6E 2Ø 3D 2Ø 5B 45 6E 74 65 72 5D
[DG5]  153: 2Ø ØA ØD ØA FF ØA ØD 2Ø 57 65 69 74 65 72 2Ø 6D
[VE5]  154: 69 74 2Ø 5B 45 6E 74 65 72 5D 2Ø 2Ø 28 6E 75 72
[ZE5]  155: 2Ø 41 6E 7A 65 69 67 65 29 2Ø FF 2Ø 44 61 74 65
[DK5]  156: 69 28 65 6E 29 2Ø 76 6F 72 68 61 6E 64 65 6E 2Ø
[LH5]  157: ØA ØD FF 2Ø 44 61 74 65 69 28 65 6E 29 2Ø 67 65
[SG5]  158: 6C 94 73 63 68 74 2Ø ØA ØD FF 2Ø 56 65 72 7A 65
[LF5]  159: 69 63 68 6E 69 73 28 73 65 29 2Ø 67 65 6C 94 73
[6A5]  16Ø: 63 68 74 2Ø ØA ØD FF 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 46 61
[FK5]  161: 6C 73 63 68 65 72 2Ø 56 69 64 65 6F 6D 6F 64 75
[195]  162: 73 2Ø 21 2Ø ØA ØD 2Ø 2Ø 2Ø 2Ø 2Ø 45 72 6C 61 75
[5E5]  163: 62 74 65 2Ø 4D 6F 64 69 3A 2Ø 32 2C 2Ø 33 2Ø 75
[TG5]  164: 6E 64 2Ø 37 2Ø ØA ØD 38 3Ø 78 32 35 2F 6D 6F 6E
[PF5]  165: 6F 63 68 72 6F 6D 2Ø 75 6E 64 2Ø 38 3Ø 78 32 35
[AK5]  166: 2F 31 36 2Ø 46 61 72 62 65 6E ØA ØD FF E8 DF Ø2
[9S5]  167: 72 6A BE 8Ø ØØ AC 8A C8 32 ED FE C9 8Ø F9 Ø3 7E
[YX5]  168: 4F 46 8B FE Ø3 F9 B8 5C 2A 1E Ø7 AB B8 2E 2A AB
[ØU5]  169: 2B FE 8B CF 8D 3E C7 Ø1 AC 3C 61 72 Ø6 3C 7A 77
[4L5]  17Ø: Ø2 24 DF AA E2 F2 32 CØ AA E8 2E Ø1 E8 31 ØØ 83
[3M5]  171: 3E CE Ø9 ØØ 74 26 32 E4 CD 16 3D ØD 1C 75 Ø9 E8
[SK5]  172: 8D Ø2 E8 77 ØØ EB 15 9Ø 3D 1B Ø1 75 E9 EB ØD 9Ø
[7R5]  173: 8D 36 D6 Ø9 C6 Ø6 DØ Ø9 FF E8 63 Ø2 B4 4C CD 21
[LH5]  174: B4 4E B9 13 ØØ 8D 16 65 Ø1 CD 21 72 12 B4 4F CD
[4M5]  175: 21 72 2Ø 8Ø 3E 9E ØØ 2E 74 F3 E8 Ø4 Ø1 EB EE 3D
[3L5]  176: 12 ØØ 74 21 8D 36 F9 Ø9 C6 Ø6 DØ Ø9 FF E8 2F Ø2
[YS5]  177: EB 29 9Ø E8 89 Ø1 81 3E CA Ø9 FF ØØ 75 C2 8B 3E
[RU5]  178: D3 Ø9 BØ FF AB 8Ø 3E C7 Ø9 FF 74 ØF E8 A1 Ø1 C6
[YN5]  179: Ø6 DØ Ø9 FF 8D 36 1E ØA E8 Ø4 Ø2 C3 C7 Ø6 CE Ø9
[ZE5]  18Ø: ØØ ØØ C7 Ø6 CA Ø9 ØØ ØØ C7 Ø6 C8 Ø9 ØØ ØØ C6 Ø6
[PW5]  181: C7 Ø9 FF E8 84 ØØ E8 87 FF B8 ØA ØE CD 1Ø BØ ØD
[1M5]  182: CD 1Ø B8 2Ø ØE CD 1Ø 8B 16 C8 Ø9 E8 34 Ø2 8D 36
[3R5]  183: C3 ØA C6 Ø6 DØ Ø9 FF E8 C5 Ø1 E8 Ø1 ØØ C3 8D 3E
[NL5]  184: 65 Ø1 83 C7 Ø3 32 CØ AA B4 3B 8D 16 65 Ø1 CD 21
[YS5]  185: 8B 3E D3 Ø9 8B 1E CE Ø9 83 EF Ø5 32 CØ AA BØ 3A
[7T5]  186: B9 4Ø ØØ FD F2 AE FC E3 3Ø 8B D7 8A ØE 95 ØØ 8Ø
[5M5]  187: E1 FC B8 Ø1 43 CD 21 B4 3A CD 21 4B 83 FB ØØ 75
[LT5]  188: D7 B8 2Ø ØE CD 1Ø 8B 16 CE Ø9 E8 D5 Ø1 8D 36 DA
[PM5]  189: ØA C6 Ø6 DØ Ø9 FF E8 66 Ø1 C3 8D 36 C7 Ø1 8D 3E
[ZH5]  19Ø: Ø3 Ø1 A4 8Ø 7C FF ØØ 75 F9 89 36 D1 Ø9 89 36 D3
[NK5]  191: Ø9 8D 36 Ø3 Ø1 8D 3E 65 Ø1 A4 8Ø 7C FF ØØ 75 F9
[SL5]  192: C3 8D 36 9E ØØ 8D 3E 65 Ø1 B9 62 ØØ 32 CØ F2 AE
[2L5]  193: 83 EF Ø4 A4 8Ø 7C FF ØØ 75 F9 8D 16 65 Ø1 8A ØE
[YK5]  194: 95 ØØ F6 C1 1Ø 75 14 8Ø 3E C7 Ø9 FF 75 Ø3 E8 11
[TN5]  195: ØØ FF Ø6 C8 Ø9 E8 35 ØØ EB Ø7 9Ø E8 3F ØØ E8 2C
[BR5]  196: ØØ C3 B8 Ø1 43 8Ø E1 FC CD 21 B8 2Ø ØE CD 1Ø 8D
[YF5]  197: 36 65 Ø1 C6 Ø6 DØ Ø9 ØØ E8 E4 ØØ B8 ØA ØE CD 1Ø
[SG5]  198: BØ ØD CD 1Ø B4 41 8D 16 65 Ø1 CD 21 C3 8D 36 Ø3
[ØR5]  199: Ø1 8D 3E 65 Ø1 A4 8Ø 7C FF ØØ 75 F9 C3 FF Ø6 CA
[CL5]  2ØØ: Ø9 8B 3E D1 Ø9 8D 36 65 Ø1 A4 8Ø 3C ØØ 75 FA B8
[CW5]  2Ø1: 5C 2A AB B8 2E 2A AB 32 CØ AA 89 3E D1 Ø9 C3 8B
[RL5]  2Ø2: 1E CA Ø9 3B 1E CE Ø9 74 19 8B 36 D3 Ø9 8D 3E Ø3
[NW5]  2Ø3: Ø1 A4 8Ø 7C FF ØØ 75 F9 89 36 D3 Ø9 E8 AE FF EB
[XS5]  2Ø4: ØA 9Ø C7 Ø6 CA Ø9 FF ØØ EB Ø1 9Ø FF Ø6 CE Ø9 C3
[GK5]  2Ø5: 8D 36 C7 Ø1 8Ø 3E D5 Ø9 12 74 22 B8 2Ø ØE CD 1Ø
[5M5]  2Ø6: C6 Ø6 DØ Ø9 ØØ E8 57 ØØ B8 ØA ØE CD 1Ø BØ ØD CD
[8N5]  2Ø7: 1Ø FE Ø6 D5 Ø9 8Ø 3C FF 75 DA EB 22 9Ø 56 C6 Ø6
[EM5]  2Ø8: D5 Ø9 ØØ 8D 36 85 ØA C6 Ø6 DØ Ø9 FF E8 3Ø ØØ 5E
[NN5]  2Ø9: 32 E4 CD 16 3D ØD 1C 75 F7 E8 33 ØØ EB B6 B8 ØA
[7M5]  21Ø: ØE CD 1Ø BØ ØD CD 1Ø BØ 2Ø CD 1Ø 8B 16 C8 Ø9 E8
[FM5]  211: 7Ø ØØ 8D 36 AB ØA C6 Ø6 DØ Ø9 FF E8 Ø1 ØØ C3 5Ø
[KN5]  212: B4 ØE AC 3A Ø6 DØ Ø9 74 Ø4 CD 1Ø EB F5 58 C3 5Ø
[LD5]  213: 53 51 52 Ø6 57 B4 ØF CD 1Ø 3C Ø7 74 13 3C Ø3 74
[WF5]  214: Ø7 3C Ø2 74 Ø3 EB 27 9Ø B8 ØØ B8 B6 17 EB Ø6 9Ø
```

Listing 1. »direx.com« beseitigt ein Verzeichnis mit allen zugehörigen Unterverzeichnissen und Dateien

```
[YN5]  215: B8 ØØ BØ B6 Ø7 8E CØ 33 FF B9 ØØ Ø8 8A E6 BØ 2Ø
[8N5]  216: F2 AB B4 Ø2 32 FF BA ØØ Ø1 CD 1Ø EB ØE 9Ø 8D 36
[ZW5]  217: F7 ØA C6 Ø6 DØ Ø9 FF E8 A5 FF F9 5F Ø7 5A 59 5B
[AL5]  218: 58 C3 5Ø 51 52 56 8B C2 BE ØA ØØ 33 C9 33 D2 F7
[1P5]  219: F6 52 41 ØB CØ 75 F6 58 E8 Ø7 ØØ E2 FA 5E 5A 59
[LF5]  22Ø: 58 C3 5Ø 3C ØA 73 Ø5 Ø4 3Ø EB Ø3 9Ø Ø4 37 B4 ØE
[4D1]  221: CD 1Ø 58 C3
```

Listing 1. »direx.com« beseitigt ein Verzeichnis mit allen zugehörigen Unterverzeichnissen und Dateien

```
[MH1]    1: comment|
[YE3]    2: Programm: direx.asm
[GV6]    3: Funktion: Directory und Subdirectories
[9M1]    4:  löschen
[MU5]    5:  Aufruf: direx [Laufwerk]:\[dir]
[US4]    6: Sprache: Masm ab 4.Ø, Tasm ab 1.Ø
[HD3]    7: Autor: Kurt Müller
[ZN4]    8: (c)1991 DMV Widuch GmbH & Co. KG
[ØØØ]    9:
[ØØØ]   1Ø:
[F74]   11: Assemblieren und linken:
[SS1]   12: tasm direx
[R72]   13: tlink direx/t
[S7Ø]   14: |
[SR2]   15: code          segment
[8K3]   16:             assume cs:code,ds:code
[7E4]   17: ;Anfang der DTA: PSP ab  8Øh
[625]   18: ;DTA-Adr 1eh bis 29h => Dateiname
[H91]   19:             org 9eh
[9C2]   2Ø: dtaofs          db ?
[DN4]   21: ;DTA-Adr 15h => Dateiattribut
[181]   22:             org 95h
[7S2]   23: dtaattr         db ?
[A61]   24:             org 1ØØh
[NV2]   25: start:     jmp begin
[HZ3]   26: speicher1          db 98 dup(Ø)
[LZ3]   27: speicher2          db 98 dup(Ø)
[D34]   28: speicher3          db 8ØØh dup(Ø)
[DN2]   29: loeschflag     db Ø
[GX2]   3Ø: loeschfiles    dw Ø
[MS2]   31: dirzaehler     dw Ø
[1H2]   32: dirzahl        dw Ø
[UR2]   33: dirabfrage     dw Ø
[Y72]   34: marke          db ?
[743]   35: markespeicher3 dw Ø
[8A2]   36: anfang         dw Ø
[562]   37: zeile          db Ø
[S47]   38: msg1 db 'Aufruf: direx [Laufwerk]:\[dir] '
[P72]   39: db Øah,Ødh,Øffh
[9K6]   4Ø: msg2 db Øah,Ødh,' Datei oder Pfad nicht g'
[BS3]   41: db 'efunden ',Øah,Ødh,Øffh
[3L6]   42: msg3 db Øah,Ødh,' Verzeichnis und alle Un'
[RØ6]   43: db 'terverzeichnisse löschen ? ? ? '
[AK1]   44: db   Øah,Ødh
[AE5]   45: db ' Abbrechen = [Esc]    Löschen = '
[SC4]   46: db '[Enter] ',Øah,Ødh,Øah,Øffh
[WK6]   47: msg4 db Øah,Ødh,' Weiter mit [Enter]  (nu'
[VX2]   48: db 'r Anzeige) ',Øffh
[MW4]   49: msg5 db ' Datei(en) vorhanden '
[P72]   5Ø: db Øah,Ødh,Øffh
[HP4]   51: msg6 db ' Datei(en) gelöscht '
[P72]   52: db Øah,Ødh,Øffh
[U16]   53: msg7 db ' Verzeichnis(se) gelöscht '
[P72]   54: db Øah,Ødh,Øffh
[D86]   55: falschmsg db '       Falscher Videomodus '
[UØ2]   56: db '! ',Øah,Ødh
[FV4]   57: db '     Erlaubte Modi: 2, 3 und 7 '
[9N1]   58: db  Øah,Ødh
[496]   59: db '8Øx25/monochrom und 8Øx25/16 Farben'
[P72]   6Ø: db Øah,Ødh,Øffh
[2Ø1]   61: begin:
[3E1]   62:       call bild
[F61]   63:       jc ende1
[ØN1]   64:       mov si,8Øh
[EK5]   65: ;Offset der Kommandozeile im PSP
[CVØ]   66:       lodsb
[VØ7]   67: ;Länge der Kommandozeile in al mit 'enter'
[KF1]   68:       mov cl,al
[6F1]   69:       xor ch,ch
[DZØ]   7Ø:       dec cl
[7S4]   71: ;wegen 'enter' 1 subtrahieren
[U81]   72:       cmp cl,3
[3K4]   73: ;Minimaleingabe = 4 Zeichen
[786]   74: ;Abfrage verhindert Eingabe von z.B.:
[ZW5]   75: ;direx c:\  = Löschen der Festplatte
[LV1]   76:       jle syntax
[911]   77:       inc si
[RF1]   78:       mov di,si
[4C1]   79:       add di,cx
```

```
[ØT4]   8Ø: ;wildcard '*.*' wird angefügt
[4V1]   81:       mov ax,'*\'
[BC1]   82:       push ds
[G31]   83:       pop es
[ZYØ]   84:       stosw
[UP1]   85:       mov ax,'*.'
[ZYØ]   86:       stosw
[5E1]   87:       sub di,si
[LG1]   88:       mov cx,di
[1N2]   89:       lea di,speicher3
[BG6]   9Ø: ;Umwandlung Klein- in Großbuchstaben
[DE1]   91: wandeln:
[CVØ]   92:       lodsb
[XR5]   93: ;Zeichen aus Kommandozeile nach al
[1G1]   94:       cmp al,'a'
[AM2]   95:       jb nichtwandeln
[LH1]   96:       cmp al,'z'
[7M2]   97:       ja nichtwandeln
[WU1]   98:       and al,Ødfh
[L82]   99: nichtwandeln:
[NXØ]  1ØØ:       stosb
[5X5]  1Ø1: ;Zeichen in al nach ES:DI = speicher3
[X32]  1Ø2:       loop wandeln
[MF1]  1Ø3:       xor al,al
[NXØ]  1Ø4:       stosb
[UY2]  1Ø5: ;Null als Endmarke
[NT2]  1Ø6:       call uebertragen
[PW1]  1Ø7:       call haupt1
[NP2]  1Ø8:       cmp dirabfrage,Ø
[N61]  1Ø9:       je ende1
[FA1]  11Ø: endlos:
[ØF1]  111:       xor ah,ah
[G81]  112:       int 16h
[CF6]  113: ;Funktion Ø wartet auf 'enter' oder 'esc'
[S53]  114:       cmp ax,1cØdh ;enter
[ZE1]  115:       jne fall1
[3E1]  116:       call bild
[UW1]  117:       call haupt2
[6G1]  118:       jmp ende1
[GG4]  119: fall1:      cmp ax,Ø11bh ;escape
[TS1]  12Ø:       jne endlos
[6G1]  121:       jmp ende1
[ZU6]  122: ;Fehler bei Eingabe:Syntaxausgabe=>Ende
[RK3]  123: syntax:     lea si,msg1
[7B2]  124:       mov marke,Øffh
[LK2]  125:       call writetext
[YR2]  126: ende1:      mov ah,4ch
[381]  127:       int 21h
[ØØØ]  128:
[H46]  129: ;Haupt1 sucht dir,subdirs,schreibt in
[WS6]  13Ø: ;speicher1,2,3,zeigt Inhalt von Speicher3
[ØØØ]  131:
[AC3]  132: haupt1          proc near
[KL6]  133: ;Dos-Funktion 4eh (finde ersten Eintrag)
[9V6]  134: ;sucht nach dem angegebenen Verzeichnis
[WW6]  135: ;Das Attribut in cx ist bitweise codiert
[547]  136: ;Unterverzeichnis=1Øh,read-only=1,hidden=2
[FSØ]  137: top:
[ZM1]  138:       mov ah,4eh
[CN1]  139:       mov cx,13h
[DP2]  14Ø:       lea dx,speicher2
[381]  141:       int 21h
[HH1]  142:       jc fehler1
[6T1]  143: naechster:
[R67]  144: ;Dos-Funktion 4fh=>finde nächsten Eintrag
[EN6]  145: ;sucht weitere Einträge im Verzeichnis
[1N1]  146:       mov ah,4fh
[381]  147:       int 21h
[981]  148:       jc holen
[S67]  149: ;keine weiteren Dateien=>carryflag gesetzt
[DT4]  15Ø:       cmp byte ptr ds:[dtaofs],'.'
[YY1]  151:       je naechster
[C36]  152: ;gefundener Eintrag =>DTA schreiben
[WØ2]  153:       call dtapfad
[LA2]  154:       jmp naechster
[5A1]  155: fehler1:
[8D1]  156:       cmp ax,18
[UH1]  157:       je fehler2
[CU1]  158:       lea si,msg2
[7B2]  159:       mov marke,Øffh
[LK2]  16Ø:       call writetext
[8G1]  161:       jmp ende2
[ZN3]  162: holen:      call nextdir
[DB3]  163:       cmp dirzaehler,Øffh
[7C1]  164:       jne top
[152]  165:       mov di,anfang
[3Y1]  166:       mov al,Øffh
[ZYØ]  167:       stosw
[9A1]  168: fehler2:
[G93]  169:       cmp loeschflag,Øffh
[P61]  17Ø:       je ende2
[Ø12]  171:       call anzeige
[7B2]  172:       mov marke,Øffh
[GU1]  173:       lea si,msg3
[LK2]  174:       call writetext
[LV1]  175: ende2:      ret
```

```
[6H2]  176: haupt1          endp
[ØØØ]  177:
[CP6]  178: ;setzt das Löschflag,ruft Haupt1 auf u.
[YE3]  179: ;gibt Endemeldung aus
[DC3]  18Ø: haupt2          proc near
[7S2]  181:       mov dirabfrage,Ø
[1T2]  182:       mov dirzaehler,Ø
[9Z2]  183:       mov loeschfiles,Ø
[2B3]  184:       mov loeschflag,Øffh
[NT2]  185:       call uebertragen
[PW1]  186:       call haupt1
[FY1]  187:       mov ax,Øe0ah
[Y71]  188:       int 1Øh
[5N1]  189:       mov al,Ødh
[Y71]  19Ø:       int 1Øh
[HW1]  191:       mov ax,Øe2Øh
[Y71]  192:       int 1Øh
[W63]  193:       mov dx,loeschfiles
[NY2]  194:       call writedecimal
[WP5]  195: ;Ausgabe: Anzahl gelöschter files
[WU1]  196:       lea si,msg6
[7B2]  197:       mov marke,Øffh
[LK2]  198:       call writetext
[WY1]  199:       call deldir
[3PØ]  2ØØ:       ret
[9H2]  2Ø1: haupt2          endp
[ØØØ]  2Ø2:
[4T5]  2Ø3: ;Proc entfernt read-only u. hidden,
[9W6]  2Ø4: ;falls vorhanden u. löscht alle subdirs
[BD3]  2Ø5: deldir          proc near
[ØN2]  2Ø6:       lea di,speicher2
[461]  2Ø7:       add di,3
[MF1]  2Ø8:       xor al,al
[NXØ]  2Ø9:       stosb
[XL6]  21Ø: ;Dos-Funktion 3bh wechselt das aktuelle
[T52]  211: ;Verzeichnis
[SM1]  212:       mov ah,3bh
[DP2]  213:       lea dx,speicher2
[381]  214:       int 21h
[152]  215:       mov di,anfang
[4X2]  216:       mov bx,dirabfrage
[BD1]  217: delring:
[N81]  218:       sub di,5
[MF1]  219:       xor al,al
[NXØ]  22Ø:       stosb
[5H1]  221:       mov al,':'
[VE1]  222:       mov cx,64
[KNØ]  223:       std
[712]  224:       repnz scasb
[ULØ]  225:       cld
[7Ø2]  226:       jcxz delend
[NG1]  227:       mov dx,di
[RØ5]  228:       mov cl,byte ptr ds:[dtaattr]
[TK6]  229: ;Attribut read-only und hidden löschen
[562]  23Ø:       and cl,111111ØØb
[FV1]  231:       mov ax,43Ø1h
[381]  232:       int 21h
[426]  233: ;Dos-Funktion 3ah löscht ein subdir
[PM1]  234:       mov ah,3ah
[381]  235:       int 21h
[FØ1]  236:       dec bx
[T91]  237:       cmp bx,Ø
[312]  238:       jne delring
[HW1]  239:       mov ax,Øe2Øh
[Y71]  24Ø:       int 1Øh
[Z86]  241: ;Ausgabe: Anzahl gelöschter subdirs
[8X2]  242:       mov dx,dirabfrage
[NY2]  243:       call writedecimal
[ØV1]  244:       lea si,msg7
[7B2]  245:       mov marke,Øffh
[LK2]  246:       call writetext
[Z92]  247: delend:      ret
[YM2]  248: deldir          endp
[ØØØ]  249:
[5U6]  25Ø: ;Inhalt speicher3=>speicher1=>speicher2
[WØ4]  251: uebertragen     proc near
[ZN2]  252:       lea si,speicher3
[ZM2]  253:       lea di,speicher1
[MF1]  254: schreib2:
[PXØ]  255:       movsb
[E54]  256:       cmp byte ptr ds:[si-1],Ø
[K12]  257:       jne schreib2
[9M3]  258:       mov markespeicher3,si
[A62]  259:       mov anfang,si
[XN2]  26Ø:       lea si,speicher1
[ØN2]  261:       lea di,speicher2
[LF1]  262: schreib1:
[PXØ]  263:       movsb
[E54]  264:       cmp byte ptr ds:[si-1],Ø
[H12]  265:       jne schreib1
[3PØ]  266:       ret
[S53]  267: uebertragen     endp
[ØØØ]  268:
[KP6]  269: ;Eintrag aus DTA in speicher2 schreiben
[TL3]  27Ø: dtapfad         proc near
[L42]  271:       lea si,dtaofs
[ØN2]  272:       lea di,speicher2
```

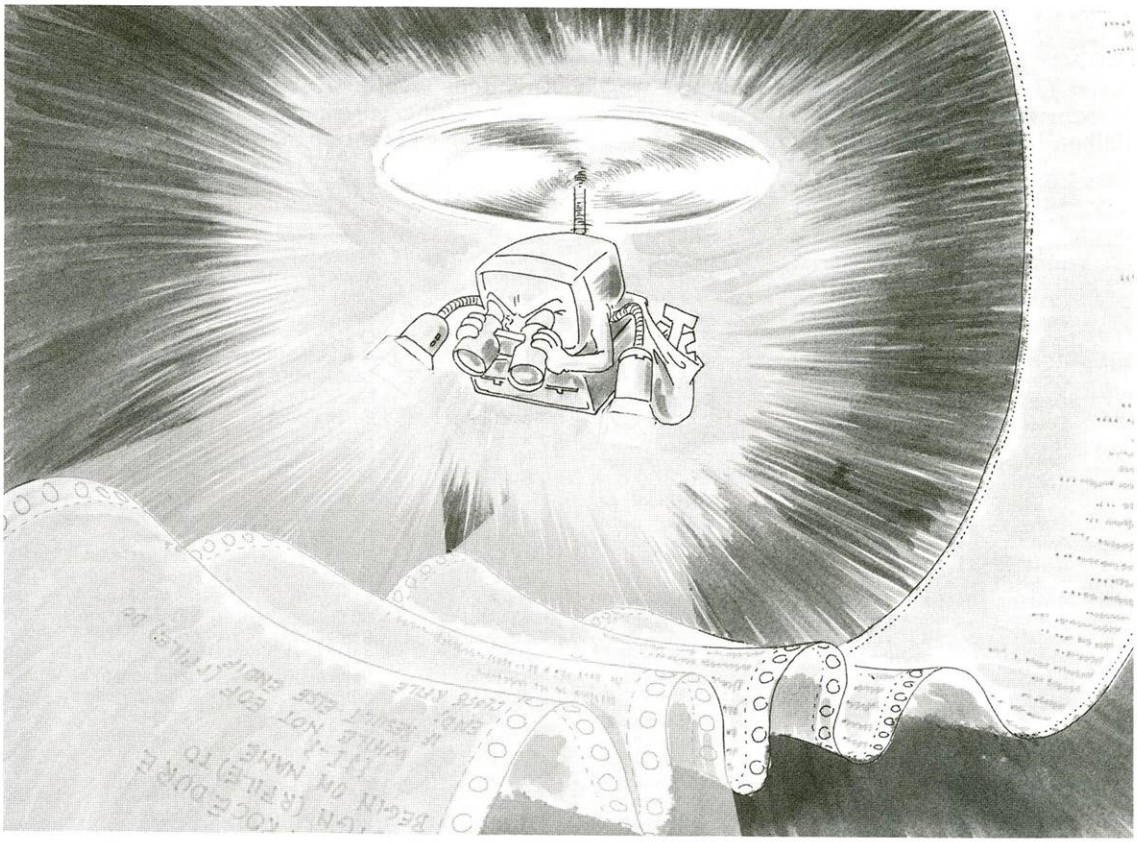
```
[6F1]  273:       mov cx,98
[7A6]  274: ;nach endmarke 'Ø' suchen, dann eintragen
[MF1]  275:       xor al,al
[712]  276:       repnz scasb
[M81]  277:       sub di,4
[ZA3]  278: ;wildcard entfernen
[8C1]  279: schreib:
[PXØ]  28Ø:       movsb
[E54]  281:       cmp byte ptr ds:[si-1],Ø
[ØØ2]  282:       jne schreib
[DP2]  283:       lea dx,speicher2
[RØ5]  284:       mov cl,byte ptr ds:[dtaattr]
[EW1]  285:       test cl,1Øh
[1V5]  286: ;überprüfen: directory oder file
[XA1]  287:       jne dir
[G93]  288:       cmp loeschflag,Øffh
[WC6]  289: ;erst nach Anzeige der gefundenen dir u.
[G57]  29Ø: ;subdir kann das Löschflag gesetzt werden
[Y87]  291: ;und procedure 'loeschen' aufgerufen werden
[EC1]  292:       jne file
[U62]  293:       call loeschen
[ØT3]  294: file:       inc loeschfiles
[U82]  295: ;Files zählen
[M94]  296:       call speicher1tospeicher2
[GS1]  297:       jmp dtaend
[ØC5]  298: dir:       call speicher2tospeicher3
[M94]  299:       call speicher1tospeicher2
[ØA2]  3ØØ: dtaend:       ret
[2S2]  3Ø1: dtapfad          endp
[ØØØ]  3Ø2:
[EC3]  3Ø3: loeschen       proc near
[US6]  3Ø4: ;Funktion 43Ø1h ändert das Dateiattribut
[FV1]  3Ø5:       mov ax,43Ø1h
[562]  3Ø6:       and cl,111111ØØb
[YT6]  3Ø7: ;Attribute read-only u. hidden 'löschen'
[381]  3Ø8:       int 21h
[HW1]  3Ø9:       mov ax,Øe2Øh
[Y71]  31Ø:       int 1Øh
[YN2]  311:       lea si,speicher2
[XX1]  312:       mov marke,Ø
[LK2]  313:       call writetext
[FY1]  314:       mov ax,Øe0ah
[Y71]  315:       int 1Øh
[5N1]  316:       mov al,Ødh
[Y71]  317:       int 1Øh
[2Ø5]  318: ;Funktion 41h löscht die Files
[RL1]  319:       mov ah,41h
[DP2]  32Ø:       lea dx,speicher2
[381]  321:       int 21h
[3PØ]  322:       ret
[RG2]  323: loeschen       endp
[ØØØ]  324:
[N46]  325: ;Eintrag von Speicher1 nach Speicher2
[NH5]  326: speicher1tospeicher2       proc near
[XN2]  327:       lea si,speicher1
[ØN2]  328:       lea di,speicher2
[UG1]  329: schreibx:
[PXØ]  33Ø:       movsb
[E54]  331:       cmp byte ptr ds:[si-1],Ø
[6R5]  332: ;schreiben bis endmarke 'Ø' gefunden
[S22]  333:       jne schreibx
[3PØ]  334:       ret
[YN4]  335: speicher1tospeicher2       endp
[ØØØ]  336:
[TS6]  337: ;wie oben,wildcard wird wieder angefügt
[SH5]  338: speicher2tospeicher3       proc near
[AF2]  339:       inc dirzaehler
[5L3]  34Ø:       mov di,markespeicher3
[YN2]  341:       lea si,speicher2
[NF1]  342: schreib3:
[PXØ]  343:       movsb
[DV3]  344:       cmp byte ptr ds:[si],Ø
[L12]  345:       jne schreib3
[4V1]  346:       mov ax,'*\'
[ZYØ]  347:       stosw
[UP1]  348:       mov ax,'*.'
[ZYØ]  349:       stosw
[MF1]  35Ø:       xor al,al
[NXØ]  351:       stosb
[UL3]  352:       mov markespeicher3,di
[3PØ]  353:       ret
[1P4]  354: speicher2tospeicher3       endp
[ØØØ]  355:
[ØY6]  356: ;schreibt speicher3=>speicher1=>speicher2
[R77]  357: ;ermittelt letztes subdir u.setzt Endmarke
[CS3]  358: nextdir          proc near
[B13]  359:       mov bx,dirzaehler
[KV2]  36Ø:       cmp bx,dirabfrage
[RL1]  361:       je ausgabe
[XX4]  362:       mov si,word ptr ds:[anfang]
[ZM2]  363:       lea di,speicher1
[PF1]  364: schreib4:
[PXØ]  365:       movsb
[E54]  366:       cmp byte ptr ds:[si-1],Ø
[M12]  367:       jne schreib4
```

Listing 2. Das Assembler-Programm »direx.asm« kann mit MASM oder TASM assembliert werden

```
[A62]  368:       mov anfang,si
[M94]  369:       call speicher1tospeicher2
[KK2]  37Ø:       jmp nextdirend
[AD1]  371: ausgabe:
[NK4]  372: ;letztes Unterverzeichnis
[ZC3]  373:       mov dirzaehler,Øffh
[KK2]  374:       jmp nextdirend
[Z22]  375: nextdirend:
[GE2]  376:       inc dirabfrage
[3PØ]  377:       ret
[NW2]  378: nextdir          endp
[ØØØ]  379:
[H17]  38Ø: ;Der Inhalt von Speicher3 wird ausgegeben.
[ZC6]  381: ;Bei Zeile 18 wird die Ausgabe angehalten
[GD6]  382: ;und auf die Eingabe [Enter] gewartet.
[PP3]  383: anzeige          proc near
[ZN2]  384:       lea si,speicher3
[T73]  385: rund:      cmp zeile,18
[B81]  386:       je stop
[HW1]  387:       mov ax,Øe2Øh
[Y71]  388:       int 1Øh
[XX1]  389:       mov marke,Ø
[LK2]  39Ø:       call writetext
[FY1]  391:       mov ax,ØeØah
[Y71]  392:       int 1Øh
[5N1]  393:       mov al,Ødh
[Y71]  394:       int 1Øh
[5H1]  395:       inc zeile
[VA4]  396:       cmp byte ptr ds:[si],Øffh
[ZY3]  397: ;Endmarke von Speicher3
[HE1]  398:       jne rund
[XM1]  399:       jmp ausex
[2XØ]  4ØØ: stop:
[GC1]  4Ø1:       push si
[HZ1]  4Ø2:       mov  zeile,Ø
[MU1]  4Ø3:       lea si,msg4
[7B2]  4Ø4:       mov marke,Øffh
[FZ2]  4Ø5: ;Endmarke von msg4
[LK2]  4Ø6:       call writetext
[E31]  4Ø7:       pop si
[RH1]  4Ø8: anhalten:
[ØF1]  4Ø9:       xor ah,ah
[G81]  41Ø:       int 16h
[FV4]  411: ;Funktion Ø des Interrupt 16h
[EØ5]  412: ;wartet auf Tastatureingabe.
[6X1]  413:       cmp ax,1cØdh
[AX5]  414: ;Eingabe wird auf 1cØdh überprüft:
[635]  415: ;Scan- u. Ascii-Code von 'Enter'.
[N32]  416:       jne anhalten
[3E1]  417:       call bild
[UF1]  418:       jmp rund
[G31]  419: ausex:
[RK3]  42Ø:       mov ax,ØeØah ;linefeed
[Y71]  421:       int 1Øh
[FL4]  422:       mov al,Ødh ;carriage return
[Y71]  423:       int 1Øh
[8W3]  424:       mov al,2Øh ;Leerzeichen
[Y71]  425:       int 1Øh
[W63]  426:       mov dx,loeschfiles
[NY2]  427:       call writedecimal
[A66]  428: ;Anzeige der Anzahl gelöschter Files.
[SU1]  429:       lea si,msg5
[7B2]  43Ø:       mov marke,Øffh
[LK2]  431:       call writetext
[3PØ]  432:       ret
[ZU2]  433: anzeige          endp
[ØØØ]  434:
[BØ6]  435: ;Diese Proc gibt einen Text über den
[HU5]  436: ;Int 1Øh/Funkt Øeh aus.Die Adresse des
[2T6]  437: ;Text wird in DS:SI an die Proc übergeben.
[KN3]  438: writetext     proc near
[PC1]  439:       push ax
[VM1]  44Ø:       mov ah,Øeh
[XP1]  441: schreiben:
[CVØ]  442:       lodsb
[TØ2]  443:       cmp al,marke
[VØ2]  444:       je writeaus
[Y71]  445:       int 1Øh
[A82]  446:       jmp schreiben
[4M1]  447: writeaus:
[R31]  448:       pop ax
[3PØ]  449:       ret
[GT2]  45Ø: writetext     endp
[ØØØ]  451:
[XB7]  452: ;Diese Proc prüft den Videomodus,löscht
[2S6]  453: ;den Bildschirm ,setzt das Farbattribut
[2R6]  454: ;und positioniert den Cursor in 1. Zeile.
[365]  455: ;(Schneller Ersatz für scroll)
[F23]  456: bild          proc near
[PC1]  457:       push ax
[TC1]  458:       push bx
[WC1]  459:       push cx
[ZC1]  46Ø:       push dx
[EC1]  461:       push es
[3B1]  462:       push di
[XM1]  463:       mov ah,Øfh
```

```
[Y71]  464:       int 1Øh
[U81]  465:       cmp al,7
[M71]  466:       je mono
[P81]  467:       cmp al,3
[RVØ]  468:       je cga
[N81]  469:       cmp al,2
[RVØ]  47Ø:       je cga
[1V2]  471:       jmp falschmodus
[F43]  472: cga:      mov ax,Øb8ØØh
[2L6]  473: ;Startadresse des Videospeichers für
[4E5]  474: ;CGA,EGA,VGA-Karten im Textmodus
[6M1]  475:       mov dh,17h
[K15]  476: ;Farbattribut(weiß auf blau)
[P21]  477:       jmp go
[X93]  478: mono:      mov ax,ØbØØØh
[2L6]  479: ;Startadresse des Videospeichers für
[WR6]  48Ø: ;Monochrom/Herculeskarten im Textmodus
[5M1]  481:       mov dh,Ø7h
[6F2]  482: go:      mov es,ax
[EF1]  483:       xor di,di
[BU1]  484:       mov cx,2Ø48
[XE1]  485:       mov ah,dh
[ZL1]  486:       mov al,2Øh
[1R6]  487: ;Leerzeichen direkt in den Videospeicher
[852]  488:       repnz stosw
[V91]  489:       mov ah,2
[3F1]  49Ø:       xor bh,bh
[U23]  491: ;Bildschirmseite Ø
[9V1]  492:       mov dx,Ø1ØØh
[Y71]  493:       int 1Øh
[R32]  494:       jmp bildende
[Ø62]  495: falschmodus:
[1S2]  496:       lea si,falschmsg
[7B2]  497:       mov marke,Øffh
[LK2]  498:       call writetext
[ENØ]  499:       stc
[GG1]  5ØØ: bildende:
[G21]  5Ø1:       pop di
[G31]  5Ø2:       pop es
[X31]  5Ø3:       pop dx
[V31]  5Ø4:       pop cx
[T31]  5Ø5:       pop bx
[R31]  5Ø6:       pop ax
[3PØ]  5Ø7:       ret
[D62]  5Ø8: bild          endp
[ØØØ]  5Ø9:
[PS6]  51Ø: ;Diese Proc wandelt eine 16-Bit-Zahl ohne
[MS6]  511: ;Vorzeichen in DX in eine Dezimalzahl um.
[M34]  512: writedecimal     proc near
[PC1]  513:       push ax
[WC1]  514:       push cx
[ZC1]  515:       push dx
[GC1]  516:       push si
[DH1]  517:       mov ax,dx
[1E1]  518:       mov si,1Ø
[PH1]  519:       xor cx,cx
[A81]  52Ø: kreis1:
[TH1]  521:       xor dx,dx
[W21]  522:       div si
[ZC1]  523:       push dx
[P11]  524:       inc cx
[571]  525:       or ax,ax
[MP1]  526:       jne kreis1
[KVØ]  527: ring:
[R31]  528:       pop ax
[C73]  529:       call writehexdigit
[7K1]  53Ø:       loop ring
[E31]  531:       pop si
[X31]  532:       pop dx
[V31]  533:       pop cx
[R31]  534:       pop ax
[3PØ]  535:       ret
[X73]  536: writedecimal     endp
[ØØØ]  537:
[1H6]  538: ;Diese Proc wandelt den Inhalt von al in
[SP6]  539: ;eine Hexziffer um und gibt sie über die
[NZ5]  54Ø: ;Funktion Øeh des Interrupt 1Øh aus.
[1Y6]  541: ;Das Farbattribut wird von Prozedur 'bild'
[RB2]  542: ;übernommen.
[PA4]  543: writehexdigit     proc near
[PC1]  544:       push ax
[MB1]  545:       cmp al,1Ø
[NA2]  546:       jae hexletter
[ZF1]  547:       add al,3Øh
[6K2]  548:       jmp writedigit
[SV1]  549: hexletter:
[L91]  55Ø:       add al,55
[L22]  551: writedigit:
[VM1]  552:       mov ah,Øeh
[Y71]  553:       int 1Øh
[R31]  554:       pop ax
[3PØ]  555:       ret
[ME3]  556: writehexdigit     endp
[ØØØ]  557:
[B62]  558: code          ends
[4W1]  559: end     start
```

Listing 2. Das Assembler-Programm »direx.asm« kann mit MASM oder TASM assembliert werden



Turbo Pascal ab Version 4.0

Wer schöne Quellcodes will, muß nicht leiden

Spätestens dann, wenn der Quellcode von Programmen und Units ins Unermeßliche steigt, wird jedem Programmierer klar, daß er dieses Monstrum auf Dauer nur schnell und wirtschaftlich pflegen kann, wenn er sich strukturierter Darstellung bedient, sinnvolle Kommentare einfließen läßt und eine einheitliche Schreibweise einführt. Da dies sehr viel Mühe kostet, liegt es nahe, diese Bestrebungen der Bequemlichkeit wegen zu automatisieren (»Laßt den Computer arbeiten und den Menschen denken!«).

Leider kann ein Editor die Beweggründe des Programmierers nicht durchschauen und deshalb natürlich auch nicht selbständig einen Quelltext kommentieren. Anders sieht es jedoch bei einer einheitlichen Schreibweise in den Quelltexten aus. Zwar versagen auch hier die meisten Editoren ihren Dienst, doch läßt sich das Problem mit dem Hilfsprogramm »convpas.pas« (Listing) sehr einfach und schnell lösen.

Convpas wandelt einen syntaktisch richtigen Turbo-Pascal-Quelltext in die vom Benutzer gewünschte Schreibweise um. Dazu muß von Ihnen die Datei »convpas.kwd« (Tabelle) mit Hilfe eines Texteditors oder der Turbo-Pascal-Benutzeroberfläche erzeugt werden. Sie enthält die Pascal-Befehlswörter in der von Ihnen gewünschten Schreibweise. Dabei ist zu beachten, daß nur ein Befehlswort in jeder Zeile stehen darf. Außerdem müssen sich die Dateien »convpas.exe« und »convpas.kwd« im gleichen Verzeichnis befinden. Beim Abtippen ist darauf zu achten, welche Compilerversion Sie besitzen. Dementsprechend sind die nicht benötigten Zeilen zu löschen. Ist Ihre Schlüsselwortdatei korrekt generiert worden, und haben Sie unter Beachtung der Hinweise die Exe-Datei von »ConvPas« erzeugt, kann die Konvertierung starten. Die Syntax dazu lautet:

```
convpas <Eingangsdatei> <Ausgangsdatei>
```

Das Konvertierungsprogramm lädt nun die von Ihnen erzeugte Schlüsselwortdatei »convpas.kwd« in den Hauptspeicher und sortiert sie mittels der Prozedur »QSort« in alphabetischer Reihenfolge. Die als erster Parameter übergebene »<Eingabedatei>« wird in einem Stück in den Hauptspeicher geladen. Wichtig ist dabei, daß die Größe dieser Datei 64 KByte nicht übersteigt. Mehr läßt allerdings auch der Turbo-Pascal-Editor nicht zu.
Die Schlüsselworte werden isoliert und mit den in der Keyword-Tabelle befindlichen verglichen. Dabei bedient sich das Programm des sehr schnellen, »binären Suchverfahrens« (Prozedur »QSearch«). Bei erfolgreicher Suche findet ein Ersatz der Schlüsselworte durch die entsprechenden Tabelleneinträge statt, andernfalls erfolgt die standardmäßige Umwandlung in Kleinschreibung. Für die Ersatzzeichen (»*«, »(.«,».)«,»*«) werden die äquivalenten Ausdrücke

»{«, »[«, »]«, »}« eingesetzt. Hexadezimale Ausdrücke (»$1ffe«) ersetzt Convpas der besseren Lesbarkeit wegen durch großgeschriebene Ausdrücke (»$1FFE«). Selbstverständlich bleiben Kommentare und String-Konstanten unberücksichtigt, um nicht deren Inhalt zu ändern. Der so bearbeitete Quelltext wird während der Abarbeitung des Übersetzungsvorganges zeilenweise in die Datei »<Ausgabedatei>« geschrieben. (Harald Vahrenhorst/hs)

Abs	ABSOLUTE	Addr	AND
Append	Arc	ArcTan	ARRAY
Assign	AssignCrt		
Bar	Bar3d	BEGIN	BlockRead
BlockWrite	BOOLEAN	BYTE	
CASE	CHAR	ChDir	Chr
Circle	ClearDevice	ClearViewPort	Close
CloseGraph	ClrEoL	ClrScr	Concat
CONST	CONSTRUCTOR	Copy	Cos
Crt	CSeg		
Dec	Delay	Delete	DelLine
DESTRUCTOR	DetectGraph	DiskFree	DiskSize
Dispose	DIV	DO	Dos
DosExitCode	DosVersion	DOWNTO	DrawPoly
DSeg			
Ellipse	ELSE	END	EnvCount
EnvStr	EoF	EoLn	Erase
Exec	Exit	ExitProc	Exp
EXTENDED	EXTERNAL		
FALSE	FExpand	FILE	FilePos
FileSize	FileSize	FillChar	FillEllipse
FillPoly	FindFirst	FindNext	FloodFill
Flush	FOR	FORWARD	Frac
FreeMem	FSearch	FSplit	FUNCTION
GetArcCoords	GetAspectRatio	GetBkColor	GetCBreak
GetColor	GetDate	GetDefaultPalette	GetDir
GetDriverName	GetEnv	GetFAttr	GetFillPattern
GetFillSettings	GetFTime	GetGraphMode	GetImage
GetIntVec	GetLineSettings	GetMaxColor	GetMaxMode
GetMaxX	GetMaxY	GetMem	GetModeName
GetModeRange	GetPalette	GetPaletteSize	GetPixel
GetTextSettings	GetTime	GetVerify	GetViewSettings
GetX	GetY	GOTO	GotoXY
Graph	Graph3	GraphDefaults	GraphErrorMsg
GraphResult			
Halt	Hi	HighVideo	
IF	ImageSize	IMPLEMENTATION	IN
Inc	InitGraph	INLINE	Input
Insert	InsLine	InstallUserDriver	InstallUserFont
Int	INTEGER	INTERFACE	INTERRUPT
Intr	IOResult		
Keep	KeyPressed		
LABEL	Length	Line	LineRel
LineTo	Ln	Lo	LONGINT
LowVideo			
Mark	MaxAvail	Mem	MemAvail
MemW	MkDir	MOD	Move
MoveRel	MoveTo	MsDos	
New	NIL	NormVideo	NoSound
NOT			
OBJECT	Odd	OF	Ofs
OR	Ord	Output	OutText
OutTextXY	OvrClearBuf	OvrGetBuf	OvrInit
OvrInitEMS	OvrSetBuf		
PackTime	ParamCount	ParamStr	Pi
PieSlice	POINTER	Port	PortW
Pos	Pred	Printer	PROCEDURE
PROGRAM	Ptr	PutImage	PutPixel
Random	Randomize	Read	ReadKey
ReadLn	RECORD	Rectangle	RegisterBGIDriver
RegisterBGIFont	REGISTERS	Release	Rename
REPEAT	Reset	RestoreCrtMode	ReWrite
Rewrite	RmDir	Round	RunError
Sector	Seek	SeekEoF	SeekEoLn
Seg	SET	SetActivePage	SetAllPalette
SetAspectRatio	SetBkColor	SetCBreak	SetColor
SetDate	SetFAttr	SetFillPattern	SetFillStyle
SetFTime	SetGraphBufSize	SetGraphMode	SetIntVec
SetLineStyle	SetPalette	SetRGBPalette	SetTextBuf
SetTextJustify	SetTextStyle	SetTime	SetUserCharSize
SetVerify	SetViewPort	SetVisualPage	SetWriteMode
SHL	SHR	Sin	SizeOf
Sound	SPtr	Sqr	Sqrt
SSeg	Str	STRING	Succ
Swap	SwapVectors	System	
TEXT	TextAttr	TextBackGround	TextColor
TextHeight	TextMode	TextWidth	THEN
TO	TRUE	Trunc	Truncate
TYPE			
UNIT	UnpackTime	UNTIL	UpCase
USES			
Val	VAR		
WhereX	WhereY	WHILE	Window
WITH	WORD	Write	WriteLn
XOR			

Die hier dargestellten Befehlswörter bilden den Inhalt der Datei »convpas.kwd« und dienen der optischen Verbesserung von Anweisungsstrukturen in der von Borland verwendeten Schreibweise (Funktionen in Groß-/Kleinschrift, Befehle und Direktiven in Großschrift). Nach Wunsch können Sie die Einträge beliebig Ihren Vorstellungen anpassen. Sie müssen die Befehle zeilenweise, ohne Leerzeilen, eingeben.

Die Turbo-Pascal-Befehlswörter bis Version 5.5

```
[U74]    1: { Programm: convpas.pas
[YP6]    2:   Funktion: Wandelt Quelltexte in von An
[FD5]    3:    wender vorgegebene Schreibweise
[EK3]    4:   Sprache: Turbo Pascal
[B94]    5:   Autor: Harald Vahrenhorst
[P35]    6:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    7:
[ZX2]    8: PROGRAM convpas;
[ØØØ]    9:
[HA6]   1Ø: { Compilerschalter für Version 4.Ø     }
[GX4]   11: {$B-,D-,F-,I-,L+,N-,R-,S-,V-          }
[ØØØ]   12:
[SE6]   13: { Compilerschalter für Version ab 5.Ø  }
[L95]   14: {$A+,B-,D-,E-,F-,L+,N-,O-,I-,R-,S-,V-  }
[ØØØ]   15:
[Ø47]   16: { Bitte nicht benötigte Zeilen löschen }
[ØØØ]   17:
[ØØØ]   18:
[YK2]   19: {$M 4Ø96,Ø,1Ø24ØØ}
[ØØØ]   2Ø:
[S62]   21: USES Crt,Dos;
[HØ4]   22: TYPE  s        = SET OF CHAR;
[LZ4]   23:       feld     = ARRAY[1..65535] OF CHAR;
[ØØØ]   24:
[ZR5]   25: CONST separators  : s = [' ',',','.',';',':',
[TK1]   26:                          ')','-','+','*',
[AS1]   27:                          '<','>','/','=',
[Ø32]   28:                          '[',']','}','#',
[D51]   29:                          '^','@'];
[H35]   3Ø:       normals     : s = ['a'..'z','A'..'Z',
[372]   31:                          'Ø'..'9','_','$'];
[BR4]   32:       exeptions   : s = ['(',#13,'"',',',
[ETØ]   33:                          '{'];
[4C4]   34:       numerics    : s = ['Ø'..'9','$'];
[CE3]   35:       zeile       : STRING = '';
[RF3]   36:       keywd       : STRING = '';
[6Y4]   37:       links       : STRING[4] = '.*[{';
[ØF2]   38:       cr_lf       = #13#1Ø;
[RNØ]   39: VAR
```

```
[M42]   4Ø:       f           : FILE;
[ZB2]   41:       ff          : TEXT;
[7D3]   42:       groesse     : LONGINT;
[9R2]   43:       p           : POINTER;
[PE4]   44:       fe          : ^feld ABSOLUTE p;
[M73]   45:       x,y,i,j,tmax: WORD;
[LL4]   46:       kwdtable    : ARRAY[1..1ØØØ] OF
[F52]   47:                     STRING[25];
[ØØØ]   48:
[ØØØ]   49:
[263]   5Ø: PROCEDURE convmsk;
[ØØØ]   51:
[MUØ]   52: BEGIN
[1FG]   53:   WriteLn('╔═══════════════════════════════',
[268]   54:           '═══════╗',cr_lf,'║                ',
[EF3]   55:           '    ConvPas 1.Ø                ',
[YRB]   56:           '║',cr_lf,'╠═══════════════════════',
[MEB]   57:           '═══════╣',cr_lf,
[EZE]   58:           '╚═══════════════════════════════',
[1F8]   59:           '═══════╝',cr_lf,'   Syntax ',
[CØ5]   6Ø:           ': ConvPas <Infile> <Outfile>',
[FA1]   61:           cr_lf  );
[ZNØ]   62: END;
[ØØØ]   63:
[YM5]   64: FUNCTION lcase(s:STRING):STRING;
[ØØØ]   65:
[MUØ]   66: BEGIN
[DY3]   67:   FOR j:= 1 TO Ord(s[Ø]) DO
[AP2]   68:     IF s[j] < #91 THEN
[HP2]   69:       IF s[j] > #64 THEN
[4G3]   7Ø:         Mem[Seg(s):Ofs(s)+j]:=
[7A4]   71:         Mem[Seg(s):Ofs(s)+j] OR $2Ø;
[VK1]   72:   lcase:= s;
[ZNØ]   73: END;
[ØØØ]   74:
[B53]   75: PROCEDURE qsearch;
[ØØØ]   76:
[F53]   77: VAR ug,m,og : WORD;
[BU2]   78:     s       : STRING[25];
```

```
[ØØØ]   79:
[MUØ]   8Ø: BEGIN
[Ø31]   81:   ug:= Ø;
[CL2]   82:   og:= Succ(tmax);
[P31]   83:   REPEAT
[FV2]   84:     m:= (og + ug) SHR 1;
[GX3]   85:     s:= lcase(kwdtable[m]);
[VØ4]   86:     IF (m = ug) OR (m = og)  THEN
[ØWØ]   87:       Exit;
[EL2]   88:     IF keywd > s THEN
[BYØ]   89:       ug:= m
[ZRØ]   9Ø:     ELSE
[8L2]   91:       IF keywd < s THEN
[ZXØ]   92:         og:= m
[ZRØ]   93:       ELSE
[MUØ]   94:         BEGIN
[KL3]   95:           keywd:= kwdtable[m];
[ØWØ]   96:           Exit;
[ZNØ]   97:         END;
[Y22]   98:   UNTIL FALSE;
[ZNØ]   99: END;
[ØØØ]  1ØØ:
[KU4]  1Ø1: PROCEDURE qsort(ug,og:WORD);
[ØØØ]  1Ø2:
[TE2]  1Ø3: VAR i,j  : WORD;
[EV2]  1Ø4:     ms,s : STRING[25];
[ØØØ]  1Ø5:
[MUØ]  1Ø6: BEGIN
[Z51]  1Ø7:    i:= ug;
[N51]  1Ø8:    j:= og;
[ET1]  1Ø9:    IF j > i THEN
[MUØ]  11Ø:      BEGIN
[384]  111:        ms:= lcase(kwdtable[(i + j)
[Y91]  112:                SHR 1]);
[P31]  113:        REPEAT
[U85]  114:          WHILE lcase(kwdtable[i]) < ms DO
[951]  115:            Inc(i);
[395]  116:          WHILE lcase(kwdtable[j]) > ms DO
[641]  117:            Dec(j);
[N12]  118:          IF i <= j THEN
[MUØ]  119:            BEGIN
[4V2]  12Ø:              s:= kwdtable[i];
[AN4]  121:              kwdtable[i]:= kwdtable[j];
[ZU2]  122:              kwdtable[j]:= s;
[951]  123:              Inc(i);
[641]  124:              Dec(j);
[ZNØ]  125:            END;
[8X1]  126:        UNTIL i > j;
[TZ1]  127:        qsort(ug,j);
[BZ1]  128:        qsort(i,og);
[ZNØ]  129:      END;
[ZNØ]  13Ø: END;
[ØØØ]  131:
[LW4]  132: PROCEDURE cperror(s:STRING);
[ØØØ]  133:
[MUØ]  134: BEGIN
[ØH4]  135:   WriteLn('** Error : ',s,' **');
[DUØ]  136:   Halt;
[ZNØ]  137: END;
[ØØØ]  138:
[834]  139: PROCEDURE loadkwdtable;
[ØØØ]  14Ø:
[MUØ]  141: BEGIN
[W91]  142:   tmax:= Ø;
[ØY3]  143:   Assign(ff,'CONVPAS.KWD');
[5M1]  144:   Reset(ff);
[5A3]  145:   IF IOResult <> Ø THEN
[EK5]  146:     cperror('ConvPas.Kwd not found');
[523]  147:   WHILE NOT EoF(ff) DO
[MUØ]  148:     BEGIN
[YM1]  149:       Inc(tmax);
[4E4]  15Ø:       ReadLn(ff,kwdtable[tmax]);
[ZNØ]  151:     END;
[WK1]  152:   Close(ff);
[Ø66]  153:   Write(tmax,' Keywords found in File ',
[6H2]  154:         'ConvPas.Kwd');
[792]  155:   qsort(1,tmax);
[A64]  156:   WriteLn(' and sorted ...');
[ZNØ]  157: END;
[ØØØ]  158:
[LH3]  159: PROCEDURE schreiben;
[ØØØ]  16Ø:
[MUØ]  161: BEGIN
[D91]  162:   Inc(i,2);
[F13]  163:   WriteLn(ff,zeile);
[ER1]  164:   zeile:= '';
[ZNØ]  165: END;
[ØØØ]  166:
[MUØ]  167: BEGIN
[646]  168:   SetCBreak(FALSE);{ Schnellere Datei-
[HL3]  169:                    zugriffe, nur TP 5.x }
[WV4]  17Ø:   { für TP 4.Ø bitte löschen ! }
[ØØØ]  171:
[VF1]  172:   convmsk;
[Z72]  173:   loadkwdtable;
[ØØØ]  174:
[VL4]  175:   IF Pos('.',ParamStr(1)) = Ø THEN
[5B4]  176:     Assign(f,ParamStr(1)+'.pas ')
[ZRØ]  177:   ELSE
[LW3]  178:     Assign(f,ParamStr(1)+' ');
[HT1]  179:   Reset(f,1);
[5A3]  18Ø:   IF IOResult <> Ø THEN
[195]  181:     cperror('InputFile not found !');
[NM3]  182:   groesse:= FileSize(f);
[HD3]  183:   IF groesse > 65535 THEN
[NC4]  184:     cperror('InputFile > 64k !');
[P13]  185:   GetMem(p,groesse);
[ZX2]  186:   Reset(f,groesse);
[AX2]  187:   BlockRead(f,p^,1);
[KD1]  188:   Close(f);
[154]  189:   Assign(ff,ParamStr(2)+' ');
[812]  19Ø:   Rewrite(ff);
[5A3]  191:   IF IOResult <> Ø THEN
[556]  192:     cperror('Cannot build Outputfile !');
[KT2]  193:   Write('Parsing ');
[7VØ]  194:   i:= 1;
[WY1]  195:   x:= WhereX;
[1Z1]  196:   y:= WhereY;
[Z53]  197:   WHILE i < groesse DO
[MUØ]  198:     BEGIN
[TZ4]  199:       WHILE fe^[i] IN separators DO
[MUØ]  2ØØ:         BEGIN
[BX3]  2Ø1:           zeile:= zeile + fe^[i];
[951]  2Ø2:           Inc(i);
[ZNØ]  2Ø3:         END;
[3U5]  2Ø4:       WHILE fe^[i] IN exeptions DO BEGIN
[LF2]  2Ø5:       CASE fe^[i] OF
[HE2]  2Ø6:         #13 : schreiben;
[PK1]  2Ø7:         '.' : BEGIN
[951]  2Ø8:                 Inc(i);
[SØ3]  2Ø9:                 IF fe^[i] = ')' THEN
[9Z2]  21Ø:                   zeile:= zeile + ']'
[ZRØ]  211:                 ELSE
[RW2]  212:                   zeile:= zeile + '.';
[ZNØ]  213:               END;
[3X1]  214:         '{' : REPEAT
[S23]  215:                 IF fe^[i] = #13 THEN
[ØR1]  216:                   schreiben;
[1U3]  217:                 IF fe^[i]+fe^[i+1] = '*)'
[2RØ]  218:                   THEN
[MUØ]  219:                   BEGIN
[FC2]  22Ø:                     fe^[i+1]:= '}';
[793]  221:                     zeile:= zeile + ' }';
[D91]  222:                     Inc(i,2);
[4MØ]  223:                   END
[ZRØ]  224:                 ELSE
[MUØ]  225:                   BEGIN
[BX3]  226:                     zeile:= zeile + fe^[i];
[951]  227:                     Inc(i);
[ZNØ]  228:                   END;
[WG3]  229:               UNTIL fe^[i-1] = '}';
[MX4]  23Ø:         '(' :  IF fe^[Succ(i)] IN ['*','.']
[2RØ]  231:                  THEN
[MUØ]  232:                  BEGIN
[DW2]  233:                    fe^[i]:= links[
[7X2]  234:                           Pos(fe^[Succ(i)],
[ØL1]  235:                           links)+2];
[4Ø3]  236:                    fe^[Succ(i)]:= #32;
[4MØ]  237:                  END
[ZRØ]  238:                ELSE
[MUØ]  239:                  BEGIN
[5W2]  24Ø:                    zeile:= zeile + '(';
[951]  241:                    Inc(i);
[ZNØ]  242:                  END;
[8K1]  243:         '''': BEGIN
[P31]  244:                 REPEAT
[BX3]  245:                   zeile:= zeile + fe^[i];
[951]  246:                   Inc(i);
[8Ø3]  247:                 UNTIL fe^[i] = '''';
[BX3]  248:                 zeile:= zeile + fe^[i];
[951]  249:                 Inc(i);
[ZNØ]  25Ø:               END;
[ZNØ]  251:       END;
[ZNØ]  252:     END;
[3L4]  253:   WHILE fe^[i] IN normals DO
[MUØ]  254:     BEGIN
[1Ø4]  255:       keywd:= keywd + fe^[i];
[951]  256:       Inc(i);
[ZNØ]  257:     END;
[ZN2]  258:   IF keywd > '' THEN
[MUØ]  259:     BEGIN
[NC3]  26Ø:       IF keywd[1] = '$' THEN
[WM4]  261:         FOR j:= 1 TO Ord(keywd[Ø]) DO
[2R4]  262:           keywd[j]:= UpCase(keywd[j])
[ZRØ]  263:       ELSE
[MUØ]  264:         BEGIN
[VF3]  265:           keywd:= lcase(keywd);
[SD1]  266:           qsearch;
[ZNØ]  267:         END;
[1K3]  268:       zeile:= zeile + keywd;
[TS1]  269:       keywd:= '';
[ZNØ]  27Ø:     END;
[FØ2]  271:   GotoXY(x,y);
[A55]  272:   Write(((i / groesse)*1ØØ.Ø):3:2,' %');
[ZNØ]  273: END;
[ØØØ]  274:
[CN5]  275:  IF zeile > '' THEN WriteLn(ff,zeile);
[WK1]  276:  Close(ff);
[D84]  277:  WriteLn('  successfull !');
[KNØ]  278: END.
```

Listing. Das Programm »convpas.pas« wandelt jeden Turbo-Pascal-Quellcode in eine einheitliche Schreibweise um



Turbo Pascal ab Version 4.0

Solo für die »Unterbrecher«

In vielen Pascal-Kursen fristet die systemnahe Programmierung ein eher kärgliches Dasein, sofern sie überhaupt erwähnt wird. Zugegeben, wer sich in die finstersten Abgründe seines Computers vorwagen will, sollte schon etwas Systemkenntnis oder Experimentierfreude mitbringen, um eigene Pascal-Programme zu neuen Höhenflügen zu bewegen.

Um die neuen Fähigkeiten der systemnahen Programmierung in Turbo Pascal auszuschöpfen, muß man sich damit vertraut machen, die DOS- oder Pascal-Systemebene zu verlassen. Dies bringt es zwangsläufig mit sich, daß Programmier- oder Eingabefehler nicht »auf höherer Ebene« erkannt und mit einer Fehlermeldung bestätigt werden, vielmehr kann es unter Umständen zum Systemabsturz kommen. Die Fehlersuche in einem maschinennahen Programm kann recht schnell zu einer zeit- und nervenintensiven Angelegenheit werden. Die reumütige Rückkehr zum wohlvertrauten Standard-Pascal-Befehlsvorrat ist oft die Folge. Dabei ist die direkte Programmierung von Hardware und Betriebssystem gar nicht so kompliziert und birgt zudem zahlreiche Vorteile. Vor allem lassen sich Steigerungen der Programm- und Ausführungsgeschwindigkeit erzielen und Routinen realisieren, die mit dem Turbo-Pascal-Befehlsvorrat nicht implementierbar sind.

Die wichtigsten Hilfsmittel, die Turbo Pascal dazu bereitstellt, sind der Zugriff auf die sogenannten »Interrupts«, das Einbinden von Maschinenspracheprogrammen, Funktionen zur Manipulation von Adressen, Prozessorregistern und Flags, der Direktzugriff auf den Arbeitsspeicher und auf die Ein-/Ausgabekanäle. Dieser Beitrag widmet sich einem der interessantesten und effizientesten der systemnahen Gebiete: dem Interrupt-Zugriff.

Interrupt-Zugriffe erhöhen die Effizienz Ihrer Programme

Grob gesprochen sind Interrupts speziell vordefinierte Unterprogramme innerhalb des PC-Systems. Viele dieser Routinen werden nach dem Einschalten des Computers in den Arbeitsspeicher geladen. Ihre Hauptaufgabe ist die Verrichtung grundlegender »Verwaltungsarbeiten«. Das Betriebssystem DOS ist beispielsweise im wesentlichen nichts anderes als eine Sammlung solcher Interrupt-Programme sowie dem zugehörigen Interpreter »command.com«. Auch viele Anweisungen höherer Programmiersprachen werden auf unterer Ebene in die entsprechenden Interrupt-Aufrufe zerlegt.

»Interrupt« ist der englische Ausdruck für Unterbrechung. Diese Bezeichnung rührt daher, daß ein gerade ablaufendes Anwenderprogramm – in der Regel nur vorübergehend und für das menschliche Wahrnehmungsvermögen oft unmerklich kurz – unterbrochen wird, um die geforderte Interrupt-Routine auszuführen.

Jeder Tastendruck löst beispielsweise eine solche Unterbrechung aus. Erst nach Auswertung des eingegebenen Zeichens durch die zuständige Interrupt-Funktion wird die eigentliche Programmausführung wieder aufgenommen. Die elementarste Form solcher Unterprogramme stellen die sogenannten BIOS-Interrupts dar. BIOS steht für »Basic Input/Output-System«. Es setzt sich größtenteils aus Dienstprogrammen zur Steuerung externer Geräte zusammen. Der BIOS-Interrupt 14hex dient beispielsweise zur Initialisierung der seriellen Schnittstelle des PC. Über der BIOS-Ebene liegen die DOS-Interrupts, die schon recht anspruchsvolle Aufgaben wie das Löschen von Dateien aus Verzeichnissen regeln. Viele dieser DOS-Interrupts greifen ihrerseits auf BIOS-Interrupts zu. Auch der Programmierer selbst kann mit den zwar sehr schnellen, aber leider von PC zu PC nicht immer kompatiblen BIOS-Interrupts arbeiten. Das PC-System stellt insgesamt 256 solcher Interrupts zur Verfügung, die von Null bis 255 durchnumeriert sind. Sie gliedern sich in folgende Interrupt-Arten:

- Hardware-Interrupts werden hardwaremäßig am INTR-Pin des Prozessors ausgelöst. Der 8086/8088 stellt acht Hardware-Interrupts zur Verfügung.

Beispiel: Der Zeitgeber des PC löst zirka 18mal pro Sekunde den Hardware-Interrupt 8 aus.

- Software-Interrupts sind durch ein Programm erzeugte, vom Programmierer gezielt ausgelöste Interrupts. Der 8086/8088 besitzt 240 verschiedene Software-Interrupts.

Beispiel: Das Betriebssystem stellt unter der Interrupt-Nummer 21hex eine Vielzahl von Funktionen zur Verfügung, auf die auch der Programmierer zurückgreifen kann. So dient beispielsweise Interrupt 21hex, Unterfunktion 2 zur Ausgabe von Zeichen.

- Prozessor- beziehungsweise »Exception«-Interrupts werden durch bestimmte prozessorinterne, verbotene Betriebszustände ausgelöst, etwa durch eine versuchte Division durch Null.

Für fast alle Programmierwünsche steht ein Interrupt bereit

Wie läßt sich nun auf einen bestimmten Interrupt zugreifen? Ganz einfach: Sie bedienen sich seiner Nummer. Jeder dieser Kennziffern ist eine bestimmte Startadresse zugeordnet, an der das gewünschte Unterprogramm im Arbeitsspeicher zu finden ist. Die Tabelle, in der Interrupt-Nummer und zugehörige Startadresse zueinander in Beziehung gesetzt werden, nennt sich Interrupt-Vektortabelle. Sie belegt das erste KByte im Arbeitsspeicher (Adressen 0 bis 03FFh).

Die computerinternen Vorgänge bei Aufruf und Abarbeitung eines Interrupts vollziehen sich etwa folgendermaßen:

Zuerst wird die Nummer der gewünschten Funktion im AH-Prozessorregister abgelegt. Sofern die Funktion noch Parameter, also Zusatzwerte, benötigt, werden diese ebenfalls in festgelegten Registern deponiert. Alle Vorbereitungen für die »Unterbrechung« sind nun getroffen. Nach Aufruf des Interrupts wird die Adresse, an der die Ausführung des gerade laufenden Anwenderprogramms unterbrochen wurde, auf den Stapel gesichert. Dieser Stapel erfüllt dabei die Funktion eines Zwischenspeichers. Aus der Vektortabelle wird die Startadresse der Interrupt-Routine ermittelt und in das CS- und das IP-Register geschrieben. Das zugehörige Unterprogramm wird nun aufgerufen und ausgeführt. Zur Rückkehr in das Pascal-Programm wird die Adresse des unterbrochenen Programms nach CS und IP restauriert und die Ausführung dieses Programms wieder aufgenommen.

Interrupt 21hex, Funktion 1Chex

Mit dieser Funktion werden Informationen über die physikalische Struktur eines Datenträgers ermittelt.

Eingabe:	AH	= 1Chex
	AL	= Laufwerkscode
Ausgabe:	AL	= Anzahl der Sektoren pro Cluster
	CX	= Anzahl der Byte pro Sektor
	DX	= Gesamtzahl der Cluster

Anmerkung:
Bei der Laufwerkscode-Angabe ist unbedingt zu beachten, daß hier die Kennziffer 0 nicht für Laufwerk A steht, sondern für das gerade aktuelle Laufwerk. Laufwerk A hat hier den Code 1, B den Code 2, C den Code 3 und so weiter.
Beispiel:
Nach dem Aufruf der Funktion enthalten die Register zum Beispiel folgende Werte: AL = 2, CX = 512 und DX = 2371. Der analysierte Datenträger befindet sich demnach in Laufwerk B; es handelt sich um eine 1,2-MByte-Diskette. Tatsächlich liefert die Funktion 1Chex sogar noch mehr Informationen zurück. Für unsere Zwecke genügen jedoch die in Listing 4 verwendeten Angaben.

Interrupt 21hex, Funktion 36hex

Mit dieser Funktion wird der auf einem Datenträger noch freie Speicherplatz ermittelt.

Eingabe:	AH	= 36hex
	DL	= Laufwerkscode
Ausgabe:	AX	= Anzahl der Sektoren pro Cluster
	BX	= Anzahl der noch verfügbaren Cluster
	CX	= Anzahl der Byte pro Sektor
	DX	= Gesamtzahl der Cluster

Anmerkungen:
Bei der Laufwerkscode-Angabe ist unbedingt zu beachten, daß hier die Kennziffer 0 nicht für Laufwerk A steht, sondern für das gerade aktuelle Laufwerk. A hat hier den Code 1, B den Code 2, C den Code 3 und so weiter.
Ein Ausgabewert von FFFFh im AX-Register steht für eine ungültige Laufwerksangabe.
Beispiel:
Nach dem Aufruf der Funktion enthalten die Register beispielsweise folgende Werte: AX = 1, BX = 468 und CX = 512. Aus diesen Daten läßt sich problemlos die noch verbleibende Speicherkapazität des Datenträgers errechnen:

Freie Kapazität = Anzahl der freien Cluster
* Anzahl Sektoren je Cluster
* Anzahl Byte je Sektor

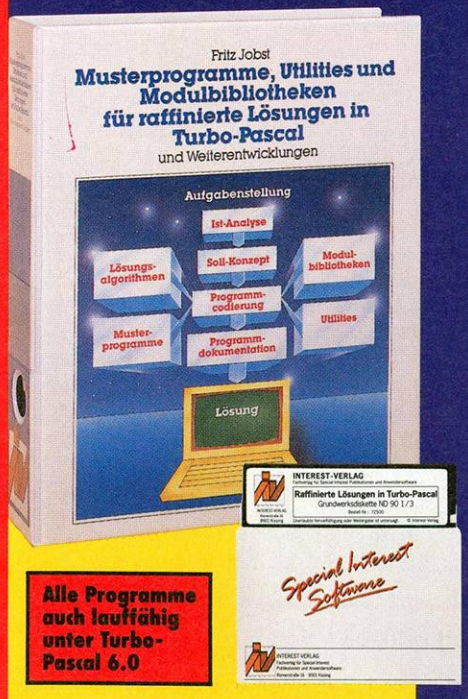
Die Turbo-Pascal-Prozedur »Intr«

Turbo Pascal bietet zwei Standardprozeduren an, mit denen sich Interrupts komfortabel aufrufen lassen:
Die Prozedur »Intr« dient zum Aufruf eines beliebigen Interrupts. Die Syntax dieses Befehls ist wie folgt:

```
Intr(<IntNo>, <Register>);
```

»IntNo« steht dabei für die jeweilige Interrupt-Nummer, kann also einen Byte-Wert zwischen Null und 255 annehmen. »Register« ist eine vom Programmierer zu vereinbarende Variable, mit der die Register des Prozessors mit den zur Ausführung benötigten Werten vorbelegt werden können. Die Variable»Register« muß vom Datentyp »Registers« sein.

PC-Bücher, die aktuell bleiben



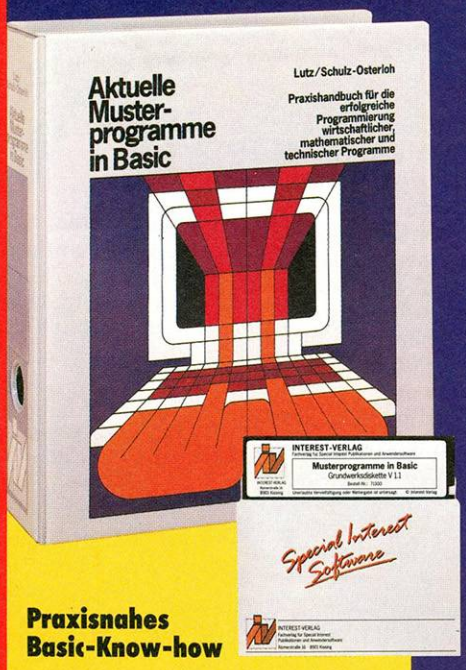
Alle Programme auch lauffähig unter Turbo-Pascal 6.0

Musterprogramme, Utilities und Modulbibliotheken in Turbo-Pascal

Dieses neue Nachschlagewerk bietet Ihnen u. a.:

- **detailliertes Know-how zum Turbo-Pascal-System;**
- **den Lehrgang Software-Engineering;**
- **Betriebssystem-Routinen:** u. a. ROM-BIOS und IBM- Hardware (BIOS-Variable, DOS-Funktionen, Druckeransteuerung), Druckertreiber;
- **hardwarenahe Programmierung:** u. a. Disk-Optimizer, Floppy-Tuning, Prefix-Segment und Environment, Expanded Memory (EMS);
- **raffinierte Utilities: Schnittstellenutilities** (u. a. Turbo-Inliner, Zugriff auf dBASE-Dateien), **Grafikutilities** (Hardcopy auf Matrix- und Laserdrucker), **Datenschutz** (u. a. Verschlüsseln von Dateien);
- **grafische Benutzeroberfläche SISGRAF:** GEM-ähnliche Benutzeroberfläche zur Einbindung in eigene Grafikprogramme oder zur Programmierung von Systemsteuerungen à la WINDOWS;
- **Datenbank-System INTERDAT** zum Entwickeln leistungsfähiger Datenbankanwendungen auf BTREE-Basis;
- **Anwenderprogramme für Technik/Mathematik und Geschäftsgrafiken** (u. a. Säulen- und Kreisdiagramme);
- **Turbo-Pascal-Modulbibliotheken,** u. a. Datum-, String- und DOS-Routinen, Mausfunktionen, EMS-Unterstützung und virtuelle Arrays;

Grundwerk ca. 600 Seiten, DIN A4, inkl. drei 5¼"-Disketten, Bestell-Nr.: 2500, Preis: DM 98,–



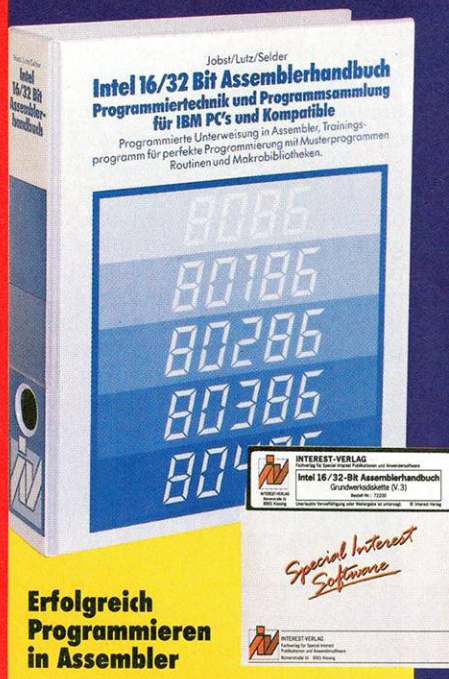
Praxisnahes Basic-Know-how

Sie erhalten u. a.

- **das menügeführte Textverarbeitungsprogramm „SOTEXT"**
- **anspruchsvolle Anwenderprogramme für folgende Bereiche:**
  - ▶ **Wirtschaft/Verwaltung** (u. a. Finanzbuchhaltung, Netzplantechnik, Terminkalender . . .);
  - ▶ **Mathematik** (u. a. Statistik, Analyse von Funktionsgleichungen, Simplex- und Gauß-Algorithmen);
  - ▶ **Technik/Physik** (u. a. Berechnen von Querschnittswerten, Wertetabellen von Digitalschaltungen);
  - ▶ **Ausbildung** (z. B. Vokabel-Training);
- **Grafik:** Grafik-Generator „SOGRAF", Zeichengenerator;
- **Dateiverwaltung und Dateiorganisation:** ein Datenbanksystem, Hash-Dateiorganisation, Such- und Sortierverfahren (Quick- und Heap-Sort);
- **Hilfsprogramme:** zum Komprimieren u. Expandieren von Textdateien, „SOZAP", ein komfortabler Disketten-Editor;
- **Drucker-Utilities** (z. B. automatische Druckereinstellung über den PC);
- **Programmierhinweise,** u. a. zu Parallelprogrammierung, Bildschirmmasken- und Menütechnik, Fehleranalyse.

Jedes Programm ist ausführlich dokumentiert mit Problembeschreibung, Vorgehensweise, Struktogramm, Listing, Probelauf sowie Anpassungshilfen für GW-Basic von Turbo-Basic- und Quick-Basic-Programmen.

Grundwerk ca. 1100 Seiten, DIN A4, inkl. 5¼"-Diskette, Bestell-Nr.: 1300, Preis: DM 98,–



Erfolgreich Programmieren in Assembler

Umsteiger wie auch versierte Praktiker erhalten mit diesem Werk praxisnahes Assembler-Wissen: Es bietet Programmierkurse, ausführliche Befehlsbeschreibungen der Intel-Prozessoren 80(x)86 und 80(x)87 sowie ausgefeilte Musterprogramme.

Aus dem Inhalt:

- **Systematische Programmierkurse** inkl. Übungsbeispielen, u. a. zur Stringverarbeitung und Interruptprogrammierung (z. B. Mausinterrupts), Assemblerkurs MASM 5.x und Testhilfe Code-View.
- **Detaillierte Befehls- und Funktionsbeschreibung** der Prozessoren 80(x)86 und 80(x)87. Sie erhalten jeweils ein Anwendungsbeispiel, die Debugsitzung, den Maschinencode und Zeitbedarf.
- **Bibliotheken,** u. a. Stringbibliothek zum Lesen, Schreiben, Kopieren, Umwandeln. Routinen zur Druckeransteuerung über das ROM-BIOS, Abbrechen der Druckerausgabe.
- **Bewährte Musterlösungen in Assembler:** Sie erhalten u. a. **Systemprogramme** zur Änderung der Fatal-Error-Interrupt-Routine und zum Komprimieren bzw. Expandieren von Textdateien, **Utilities** wie komfortable Drucker- und Steuerprogramme sowie **Rechnerkopplungsprogramme,** u. a. ein Programm zum Datentransfer zwischen PC-/MS-DOS-Rechnern, **mathematische und technische Programme,** die Programmierung des Timers 8253-5 und **Anwenderprogramme** zum Generieren und Überprüfen eines Paßwortes für Systemdisketten, EXE-Dateien-Ver- und Entschlüsseln, Sektorendump, Dateidump.
- **Grundwerks-Diskette** mit praxisbezogenen Programmen und Utilities.

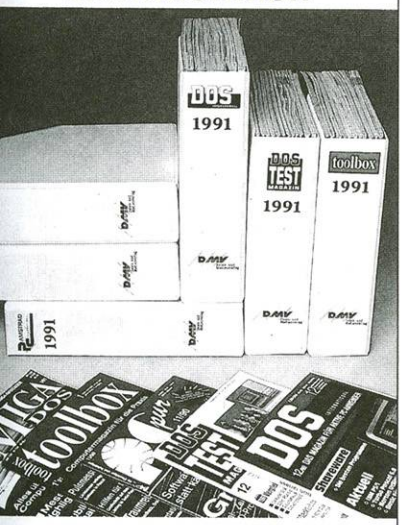
Grundwerk ca. 1000 Seiten, DIN A4, inkl. zwei 5¼"-Disketten, Bestell-Nr.: 2200, Preis: DM 98,–

INTEREST-VERLAG
Fachverlag für Special Interest Publikationen und Anwendersoftware
Römerstraße 16
D-8901 Kissing
Tel. 0 82 33 / 2 11-0

Zu jedem dieser Werke erhalten Sie alle 2 – 3 Monate Erweiterungen Ihres Grundwerks mit je ca. 130 Seiten, inkl. Diskette, zum Preis von DM 59,– (Abbestellung jederzeit möglich).

Für Ihre Anforderung verwenden Sie bitte nebenstehende Bestellkarte

Zum Sammeln unentbehrlich



Die DMV-Sammelordner

Sonderhefte • Sammelordner • Demonstrationen

Ausgabe DOS EXTRA DM 18,–	DATABOX 2 Stk. 5 1/4"-Disk DM 35,–	DATABOX 1 Stk. 3 1/2"-Disk DM 35,–
☐ 356 Nr. 3	☐ 357	☐ 358
☐ 381 Nr. 4	☐ 3811	☐ 3812
☐ 382 Nr. 5	☐ 3813	☐ 3814
☐ 383 Nr. 6	☐ 3815	☐ 3816
☐ 384 Nr. 7	☐ 3817	☐ 3818
☐ 385 Nr. 8	☐ 3819	☐ 3820
☐ 386 Nr. 9	☐ 3821	☐ 3822
☐ 387 Nr. 10	☐ 3823	☐ 3824
☐ 388 Nr. 11	☐ 3825	☐ 3826
☐ 389 Nr. 12	☐ 3827	☐ 3828
☐ 391 Nr. 13	———	———
☐ 392 Nr. 14	☐ 3829	☐ 3830
Pascal Sonderdruck	☐ 346	☐ 347
DMV-Computer-wissen Band 1	☐ 3512	☐ 3513

DOS-Taschenbücher	5 1/4"-Diskette DM 24,–	3 1/2"-Diskette DM 24,–
BAND 1	☐ 60501	☐ 60502
BAND 3	☐ 60521	☐ 60522
BAND 4	☐ 60531	☐ 60532

Best.-Nr.		DM
☐ 60500	DOS-Taschenbuch Band 1	29,80
☐ 60510	DOS-Taschenbuch Band 2	29,80
☐ 60520	DOS-Taschenbuch Band 3	29,80
☐ 60530	DOS-Taschenbuch Band 4	29,80
☐ 402	Die Basic2-Toolbox	49,–
☐ 403	Die Basic2-Toolbox, 5 1/4"-Diskette zum Buch	29,–
☐ 345	PASCAL Sonderdruck	25,–
☐ 3511	DMV-Computer-Wissen, Band 1, Basic-Toolbox	18,–
☐ 482	QuickStart ConText PRO	19,80

Sammelordner 2 Stk. 15,80

☐ 530 DOS
☐ 534 toolbox
☐ 536 DOS TEST
☐ 531 PC International
☐ 535 PCpur
☐ 537 AMIGA DOS

Zwischensumme + Porto und Verpackung
(Inland 4,– DM, Ausland 6,– DM) _____

Demodisketten jeweils DM 5,–

☐ 001 Fraktal 3D + MaskEdit
☐ 005 ConText PRO
☐ 008 DMV-Show-Manager
☐ 012 GraphBas PC
☐ 003 ERGO 3.0
☐ 007 3D-Draw
☐ 009 Navigator

Gesamtbetrag: _____ DM

PC-Software

vom

DMV Versandservice

- leistungsstark
- praxisbewährt
- preisbewußt

PC-Software

5 1/4"	3 1/2"		DM
2282 ☐	2281 ☐	**ConText EASY,**	99,–
2407 ☐	2408 ☐	**ConText PRO,**	199,–
2415 ☐	2416 ☐	**ConText-ADREVA**	49,–
2419 ☐	2420 ☐	**ConText-Zeicheneditor**	99,–
2421 ☐	2422 ☐	**ConText-Rechtschreibhilfe**	69,–
2650 ☐	2651 ☐	**DMV-Power-Pack**	349,–
256 ☐	257 ☐	**PC-VirusDoktor**	99,–
2903 ☐	2904 ☐	**PC-VirusFinder**	149,–
2905 ☐	2906 ☐	**GraphBas PC,** Turbo Basic	99,–
2907 ☐	2908 ☐	**GraphBas PC,** Microsoft Quick Basic	99,–
2522 ☐	2521 ☐	**ERGO 3.0,** DOS-Benutzeroberfläche	199,–
2702 ☐	2703 ☐	**DMV-Show-Manager**	199,–
238 ☐	239 ☐	**MaskEdit Plus** mit einem Treiber	99,–
2613 ☐	2614 ☐	**MaskEdit Plus,** Paket mit zwei Treibern	149,–
2621 ☐	2622 ☐	**MaskEdit Plus,** Paket mit drei Treibern	199,–

Turbo Pascal 3/4/5	Turbo C Microsoft C	Top-Speed Modula-2	Turbo Basic	Quick Basic
☐ 2632	☐ 2633	☐ 2634	☐ 2635	☐ 2636

2720 ☐	2721 ☐	**WinBasic**	299,–
227 ☐	233 ☐	**BCI DisAsm/I86,** (MS-DOS)	199,–
6001 ☐	6002 ☐	**RSM-Manager**	69,–
6007 ☐	6008 ☐	**3D-Draw, Version 1.0**	99,–
6017 ☐	6018 ☐	**3D-Draw Professional**	199,–
6019 ☐	6020 ☐	**Navigator 4.5**	69,–
6025 ☐	6026 ☐	**Lightning 3D, 1.5**	199,–
6027 ☐	6028 ☐	**DOS-CAD 3.0**	199,–
2700 ☐	2701 ☐	**DMV-Faktura**	398,–
2710 ☐	2711 ☐	**DMV-Atelier**	249,–
2712 ☐	2713 ☐	**DMV-Atelier** Zusatzfonts	99,–
2714 ☐	2715 ☐	**DMV-Atelier** Symbole	99,–

5 1/4"	3 1/2"		DM
20506 ☐	20507 ☐	**PC-Software pur,** Festplatten-Optimierer	69,–
20508 ☐	20509 ☐	**PC-Software pur,** MasterDat Datenbank	69,–
20510 ☐	20511 ☐	**PC-Software pur,** MS-DOS Lernprogramm	69,–
2804 ☐	2805 ☐	**toolbox Spezial II,** PASCOMP	35,–
2808 ☐	2809 ☐	**toolbox Spezial IV,** CALC	35,–
2815 ☐	2816 ☐	**toolbox Spezial VII,** dt. Silbentrennung	35,–
2819 ☐	2820 ☐	**toolbox Spezial IX,** HPGL-Plotter	35,–
2823 ☐	2824 ☐	**toolbox Spezial XI,** MAP, MARK & RELEASE	35,–
2827 ☐	2828 ☐	**toolbox Spezial XIII,** Trickkiste Vol. I	35,–
2829 ☐	2830 ☐	**toolbox Spezial XIV,** Trickkiste Vol. 2	35,–
2831 ☐	2832 ☐	**toolbox Spezial XV,** Fitting & Lin.-Glei.	35,–
2833 ☐	2834 ☐	**toolbox Spezial XVI,** nat.-sprachl. Schnittst.	35,–
2835 ☐	2836 ☐	**toolbox Spezial XVII,** EMS-Speicher-Software	35,–
2837 ☐	2838 ☐	**toolbox Spezial XVIII,** Alkane-Klassifizierung	35,–
2839 ☐	2840 ☐	**toolbox Spezial XIX,** Grafik & Spieleprogrammierung	35,–
2361 ☐	2362 ☐	**MAUSALL Plus**	35,–
1371 ☐	1372 ☐	**PC-Spiele-Sampler**	49,–
248 ☐	249 ☐	**Fraktal-Generator 3D,** MS-DOS	69,–
1374 ☐	—	**Populous PC**	69,95
1375 ☐	1376 ☐	**Bards Tale II**	39,95
1377 ☐	1378 ☐	**Starflight II**	49,95
1379 ☐	1380 ☐	**Indianapolis 500**	49,95
1381 ☐	1382 ☐	**Oil Imperium**	39,95
1383 ☐	1384 ☐	**Loom**	69,95
1385 ☐	1386 ☐	**Zak McKracken**	69,95
1387 ☐	1388 ☐	**Maniac Mansion**	69,95
1389 ☐	—	**Jet Pack**	169,–
1401 ☐	—	**Digi-Joy**	59,–
1295 ☐	1296 ☐	**Memo**	69,–
1297 ☐	1298 ☐	**Trademaster**	49,–

+ Porto/Verpackung
(Inland 4,– DM, Ausland 6,– DM) _____ DM = Gesamtbetrag: _____ DM

Ihre Anforderungskarte

Programmieren Sie Ihren Erfolg!

Prüfen Sie
das Nachschlagewerk in Ruhe zu Hause. Wenn Sie nicht überzeugt sind, schicken Sie dieses mit den Disketten innerhalb von 10 Tagen an uns zurück. Damit ist die Sache für Sie erledigt.

Ihre Anforderungskarte

Bitte abtrennen oder ausschneiden!

Ja, senden Sie mir sofort:

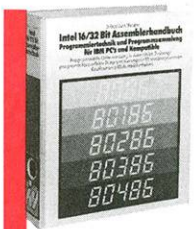
(Zutreffendes bitte ankreuzen)



☐ Expl. **„Musterprogramme, Utilities und Modulbibliotheken für raffinierte Lösungen in Turbo-Pascal"**
praktischer Ringbuchordner, DIN A4, ca. 600 Seiten, inkl. drei 5 1/4"-Disketten, Bestell-Nr.: 2500, Preis: DM 98,–, wahlweise zwei 3 1/2"-Disketten.



☐ Expl. **„Aktuelle Musterprogramme in Basic"**
stabiler Ringbuchordner, DIN A4, ca. 1100 Seiten, inkl. 5 1/4"- Diskette, Bestell-Nr.: 1300, Preis: DM 98,–, wahlweise 3 1/2"-Diskette.



☐ Expl. **„INTEL 16/32 Bit Assembler-Handbuch"**
stabiler Ringbuchordner, DIN A4, ca. 1000 Seiten, inkl. zwei 5 1/4"-Disketten, Bestell-Nr.: 2200, Preis: DM 98,–, wahlweise 3 1/2"-Diskette.

Disketten-Service
Ich wünsche die Disketten im
☐ 3 1/2"-Format (gegen jeweils DM 2,– Aufpreis)
Nicht vergessen! Geben Sie bitte umseitig ihre vollständige Adresse an.

Zu jedem dieser Werke erhalten Sie alle 2 – 3 Monate Erweiterungen Ihres Grundwerks mit je ca. 130 Seiten, inkl. Diskette, zum Preis von DM 59,– (Abbestellung jederzeit möglich).

PC-Sonderhefte

Ja, ich mache von Ihrem Angebot Gebrauch und bestelle die umseitig ausgewählten Produkte. Bitte liefern Sie an folgende Anschrift:

Name

Straße/Postfach

PLZ/ORT

☐ Den Betrag bezahle ich mit dem beigefügten Verrechnungsscheck.

☐ Ich bitte um Lieferung per Nachnahme zuzüglich der Nachnahmegebühr (nur innerhalb der BRD).

Datum Unterschrift (bei Minderjährigen des gesetzlichen Vertreters)

Bitte ausreichend frankieren

Antwortkarte

DMV-Verlag
DOS International
Postfach 250

3440 Eschwege

1. Informieren

DOS-Taschenbücher und DOS EXTRA-Sonderhefte sind eine zu verlässige Informationsquelle fü jeden PC-Besitzer. Nutzen auch Sie dieses Wissen, und vertiefer Sie es mit den dazu angebotener Programmen aus dem Disketten Service. Den PC beherrschen mi

- **DOS-EXTRA-Sonderheften**
- **DOS-Taschenbüchern**
- **DOS-Disketten-Service**

Zur Erinnerung:
Meine DOS-Sonderhefte, DOS Taschenbücher, DOS-Disketten wur den bestellt am

PC-Software

Ja, ich mache von Ihrem Angebot Gebrauch und bestelle die umseitig ausgewählten Produkte. Bitte liefern Sie an folgende Anschrift:

Name

Straße/Postfach

PLZ/ORT

☐ Den Betrag bezahle ich mit dem beigefügten Verrechnungsscheck.

☐ Ich bitte um Lieferung per Nachnahme zuzüglich der Nachnahmegebühr (nur innerhalb der BRD).

Datum Unterschrift (bei Minderjährigen des gesetzlichen Vertreters)

Bitte ausreichend frankieren

Antwortkarte

DMV-Verlag
DOS International
Postfach 250

3440 Eschwege

2. Aufrüsten

PC-Software vom DMV-Verlag is unter Kennern bereits ein Begrif und auch ein gutes Angebot fü Sie, Ihren PC mit starken und an wendergerechten Applikationen auszustatten. Von der Textver arbeitung bis zum Programmier system, beim DMV-Versand service kaufen Sie günstig ein Den PC konsequent nutzen mi PC-Software aus dem DMV-Ver sandservice

- **PC-Anwendungen**
- **PC-Applikationen**
- **PC-Entwicklungssysteme**

Zur Erinnerung:
Meine PC-Software wurde bestellt an

Meine Anschrift

Name

Vorname

Straße, Haus-Nr.

PLZ, Ort

Unterschreiben Sie hier bitte Ihre Bestellung!
Bei Minderjährigen ist die Unterschrift eines gesetzlichen Vertreters erforderlich. Ohne Ihre Unterschrift kann die Bestellung nicht bearbeitet werden.

X

Datum Unterschrift

Bitte unterschreiben Sie auch Ihre Sicherheitsgarantie,
mit der Sie folgendes zur Kenntnis nehmen: Sie haben das Recht, Ihr angefordertes Werk innerhalb von 10 Tagen ab Lieferung an den INTEREST-VERLAG GmbH, Römerstraße 16, 8901 Kissing, zurückzusenden, wobei für die Fristwahrung das Datum der Absendung genügt. Sie kommen dadurch von allen Verpflichtungen aus der Bestellung frei.

X

Datum Unterschrift

IN 110378

60 Pfennig, die sich lohnen!

Postkarte/Antwort

INTEREST-VERLAG

Fachverlag für Special Interest
Publikationen und Anwendersoftware
z. Hd. Herrn Boos

Römerstraße 16

D-8901 Kissing

Verlagsgarantie

- Sie erhalten von mir PC-Fachinformationen, von absoluten Spezialisten für Sie geschrieben.
- Ihr Nachschlagewerk können Sie in aller Ruhe 10 Tage zu Hause prüfen und bei Nichtgefallen innerhalb dieser Frist zurücksenden. Der INTEREST-VERLAG gewährt nach Zahlung des Rechnungsbetrages das Recht, die beiliegende Software zu testen und zeitlich unbeschränkt zu nutzen.
- Ihr Nachschlagewerk ist immer aktuell. Dafür sorgt unser Ergänzungsservice.
- Den Ergänzungsservice können Sie jederzeit kündigen. Sie gehen also kein Risiko ein. Darauf gebe ich Ihnen mein Wort.

Ihr INTEREST-VERLAG
Fachverlag für Special Interest
Publikationen und Anwendersoftware

Michael Boos

Michael Boos
Verlagsleiter

DOS INTERNATIONAL

Kompetenz

für

PC-Personalcomputer

im

Abonnement

deutlich

günstiger

DOS INTERNATIONAL

Information

über

PC-Personalcomputer

sammeln Sie mit

- **DOS-Einzelheften**
- **DATABOX PLUS**
  mit mehr als 700 KByte Software und Bonusprogramm

DOS INTERNATIONAL

Vorsprung

für

PC-Personalcomputer

mit

DOS EXTRA

im

Abonnement

Abo-Order DOS EXTRA

Hiermit bestelle ich **»DOS EXTRA«** für mindestens

☐ **4 Ausgaben** (vierteljährliche Erscheinungsweise)

Preise: **BRD**	4 Ausgaben 60,– DM
Auslandspreise: **Europa**	4 Ausgaben 70,– DM
Außereuropäisches Ausland	4 Ausgaben 80,– DM

Lieferung erfolgt ab der nächsterreichbaren Ausgabe.

Dauerbezug DOS EXTRA DATABOX

☐ Hiermit bestelle ich **»DOS EXTRA DATABOX«** im Dauerauftrag mit Kündigungsmöglichkeit zum vierteljährlichen Erscheinungstermin.

Achtung: Nur im Bankabbuchungsverfahren möglich!

Preise: **BRD:**	je 39,– DM
Auslandspreise: **Europa:**	je 41,– DM
Außereurop. Ausland:	je 42,– DM

Lieferung erfolgt ab der nächsterreichbaren Ausgabe.

Diskettenformat: ☐ IBM 5 1/4" ☐ IBM 3 1/2"

Lieferanschrift

☐ Geschäft ☐ Privat

Vorname, Name

Firma (nur wenn Lieferadresse)

Straße, Nr./Postfach

PLZ/Ort *(Bitte genaue Anschrift angeben!)*

Dieses Abonnement verlängert sich automatisch um 4 Ausgaben, wenn es nicht sechs Wochen vor Ablauf schriftlich gekündigt wird.

Datum 1. Unterschrift (bei Minderjährigen des gesetzlichen Vertreters)

☐ **Bequem und bargeldlos durch Bankabbuchung**

Bankleitzahl

Konto-Nr./-Inhaber

Geldinstitut

☐ **Gegen Rechnung** – zahlbar innerhalb zwei Wochen nach Erhalt. (Bitte keine Vorauszahlung leisten — Rechnung abwarten.)

Bitte unbedingt auf der Vorderseite eine zweite Unterschrift leisten!

DOS-Einzelhefte und DATABOX

Einzelheft Ausgabe 6,50 DM/Stck	DATABOX 5 1/4"-Diskette 24,– DM/Stck	DATABOX 3 1/2"-Diskette 24,– DM/Stck
– 11/87	☐	☐
– 1/88	☐	☐
– 2/88	☐	☐
☐ 3/88	☐	☐
☐ 4/88	☐	☐
☐ 5/88	☐	☐
☐ 6/88	☐	☐
☐ 7/88	☐	☐
☐ 8/88	☐	☐
☐ 9/88	☐	☐
Einzelheft 7,50 DM/Stck		
☐ 10/88	☐	☐
☐ 11/88	☐	☐
☐ 12/88	☐	☐
☐ 1/89	☐	☐
☐ 2/89	☐	☐
☐ 3/89	☐	☐
☐ 4/89	☐	☐
☐ 5/89	☐	☐
☐ 6/89	☐	☐
☐ 7/89	☐	☐
☐ 8/89	☐	☐

Einzelheft Ausgabe 7,50 DM/Stck	DATABOX PLUS DATABOX + Bonus 5 1/4" 2 Disketten 24,– DM/Stck	DATABOX PLUS DATABOX + Bonus 3 1/2"-Diskette 24,– DM/Stck
☐ 9/89	☐	☐
☐ 10/89	☐	☐
☐ 11/89	☐	☐
☐ 12/89	☐	☐
– 1/90	☐	☐
☐ 2/90	☐	☐
☐ 3/90	☐	☐
☐ 4/90	☐	☐
– 5/90	☐	☐
☐ 6/90	☐	☐
☐ 7/90	☐	☐
☐ 8/90	☐	☐
☐ 9/90	☐	☐
☐ 10/90	☐	☐
☐ 11/90	☐	☐
☐ 12/90	☐	☐
☐ 1/91	☐	☐
☐ 2/91	☐	☐
☐ 3/91	☐	☐
☐ 4/91	☐	☐

☐ Gratis-Sammelordner bei Bestellung von 6 Einzelheften

+ Porto/Verpackung (Inland 4,– DM, Ausland 6,– DM) _____ DM Gesamtbetrag _____ DM

Abo-Order DOS International

Hiermit bestelle ich **»DOS International«** für mindestens

	☐ **12 Ausgaben**	☐ **6 Ausgaben**
Preise: **BRD**	12 Ausgaben 82,50 DM	6 Ausgaben 42,– DM
Auslandspreise: **Europa**	12 Ausgaben 110,– DM	6 Ausgaben 55,– DM
Außereuropäisches Ausland	12 Ausgaben 150,– DM	6 Ausgaben 75,– DM

Lieferung erfolgt ab der nächsterreichbaren Ausgabe.

Abo-Order DOS DATABOX PLUS

Hiermit bestelle ich **»DOS DATABOX PLUS«** für mindestens

☐ **12 Ausgaben** ☐ **6 Ausgaben**

Diskettenformat: ☐ IBM 5 1/4" ☐ IBM 3 1/2"

Inlandspreise:	12 Ausgaben 300,– DM, 6 Ausgaben 150,– DM
Europa:	12 Ausgaben 320,– DM, 6 Ausgaben 160,– DM
Außereurop. Ausland:	12 Ausg. 360,– DM, 6 Ausg. 180,– DM

Lieferung erfolgt ab der nächsterreichbaren Ausgabe.

Lieferanschrift

☐ Geschäft ☐ Privat

Vorname, Name

Firma (nur wenn Lieferadresse)

Straße, Nr./Postfach

PLZ/Ort *(Bitte genaue Anschrift angeben!)*

Dieses Abonnement verlängert sich automatisch um 6 bzw. 12 Ausgaben, wenn es nicht sechs Wochen vor Ablauf schriftlich gekündigt wird.

Datum 1. Unterschrift (bei Minderjährigen des gesetzlichen Vertreters)

☐ **Bequem und bargeldlos durch Bankabbuchung**

Bankleitzahl

Konto-Nr./-Inhaber

Geldinstitut

☐ **Gegen Rechnung** – zahlbar innerhalb zwei Wochen nach Erhalt. (Bitte keine Vorauszahlung leisten — Rechnung abwarten.)

Bitte unbedingt auf der Vorderseite eine zweite Unterschrift leisten!

DOS INTERNATIONAL

Abonnement

Rechnungsanschrift

Vorname, Name

Straße, Nr./Postfach

PLZ/Ort

Widerrufsrecht

Ich weiß, daß ich diese Vereinbarung innerhalb einer Woche beim DMV-Verlag, Postfach 250, 3440 Eschwege, schriftlich widerrufen kann, wobei bereits die rechtzeitige Absendung meines Widerrufschreibens zur Fristwahrung ausreicht. Ich bestätige das durch meine zweite Unterschrift.

Datum 2. Unterschrift (bei Minderjährigen des gesetzlichen Vertreters)

Bitte ausreichend frankieren

Antwortkarte

DMV-Verlag
DOS International
Postfach 250

3440 Eschwege

1. Abonnieren

Jeden Monat neu, vermitteln Ihnen die DOS International und DOS EXTRA wichtige Informationen zu Ihrem Personal-Computer. Nutzen Sie dieses Wissen mit einem Abonnement zu Ihrem ganz persönlichen

- **Preisvorteil**
- **Informationsvorsprung**
- **Zustellkomfort**

Zur Erinnerung:
Mein persönliches Abonnement wurde bestellt am

Widerrufsrecht

Ich weiß, daß ich diese Vereinbarung innerhalb einer Woche beim DMV-Verlag, Postfach 250, 3440 Eschwege, schriftlich widerrufen kann, wobei bereits die rechtzeitige Absendung des Widerrufschreibens zur Fristwahrung ausreicht.

DOS INTERNATIONAL

Einzelhefte

Ja, ich mache von Ihrem Angebot Gebrauch und bestelle die umseitig ausgewählten Produkte. Bitte liefern Sie an folgende Anschrift:

Name

Straße/Postfach

PLZ/ORT

☐ Den Betrag bezahle ich mit dem beigefügten Verrechnungsscheck.

☐ Ich bitte um Lieferung per Nachnahme zuzüglich der Nachnahmegebühr (nur innerhalb der BRD).

Datum Unterschrift (bei Minderjährigen des gesetzlichen Vertreters)

Bitte ausreichend frankieren

Antwortkarte

DMV-Verlag
DOS International
Postfach 250

3440 Eschwege

2. Sammeln

DOS-International-Leser sind immer top-informiert. Daher sollte Ihnen kein Heft oder Sonderheft in Ihrer Sammlung fehlen. Das Eintippen von Listings und Programmen ersparen Sie sich durch unseren günstigen Disketten-Service. Immer gut beraten sind Sie mit DOS International

- **Einzelheften**
- **Sonderheften**
- **Disketten-Service**

Zur Erinnerung:
Meine Hefte und Disketten wurden bestellt am

DOS INTERNATIONAL

Abonnement

Rechnungsanschrift

Vorname, Name

Straße, Nr./Postfach

PLZ/Ort

Widerrufsrecht

Ich weiß, daß ich diese Vereinbarung innerhalb einer Woche beim DMV-Verlag, Postfach 250, 3440 Eschwege, schriftlich widerrufen kann, wobei bereits die rechtzeitige Absendung meines Widerrufschreibens zur Fristwahrung ausreicht. Ich bestätige das durch meine zweite Unterschrift.

Datum 2. Unterschrift (bei Minderjährigen des gesetzlichen Vertreters)

Bitte ausreichend frankieren

Antwortkarte

DMV-Verlag
DOS International
Postfach 250

3440 Eschwege

3. Schenken

Bei jedem Anlaß ist ein Abonnement ein Geschenk, das ankommt und Ihrem Wunsch nach »gutem Erfolg« Ausdruck verleiht. Sie brauchen nur im Feld »Lieferanschrift« die Adresse des Beschenkten einzutragen.

- **DOS International**
- **DOS EXTRA**

Zur Erinnerung:
Mein Geschenk-Abonnement wurde bestellt am

Widerrufsrecht

Ich weiß, daß ich diese Vereinbarung innerhalb einer Woche beim DMV-Verlag, Postfach 250, 3440 Eschwege, schriftlich widerrufen kann, wobei bereits die rechtzeitige Absendung des Widerrufschreibens zur Fristwahrung ausreicht.

Interrupt 17hex, Funktion 2

Mit dieser Funktion wird der aktuelle Druckerstatus abgefragt.

Eingabe: AH = 2
DX = Druckernummer
Ausgabe: AH = Druckerzustand

Anmerkungen:
Drucker 1 hat normalerweise die Druckernummer 0. Wird ein Wert ungleich 90hex zurückgeliefert, so liegt vermutlich ein Druckerfehler vor (beispielsweise ist der Drucker nicht bereit oder kein Papier mehr vorhanden).

Dieser variante Record ist in der Unit »Dos« (ab Turbo Pascal 4.0) folgendermaßen vordefiniert:

```
Type Registers = Record
Case Integer OF
0: (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags: Word);
1: (AL,AH,BL,BH,CL,CH,DL,DH: Byte)
End;
```

Eine Registervariable »regs« würde demnach folgendermaßen vereinbart werden:

```
Var Regs: Registers;
```

Voraussetzung ist allerdings das Einbinden der Unit »Dos« mit Hilfe der Uses-Anweisung zu Beginn des Programms. Als Beispiel für die Verwendung von »Intr« soll Listing 1 (»cursor.pas«) dienen, mit dessen Hilfe die Cursorgröße verändert werden kann. Dazu bedienen wir uns der BIOS-Funktion 10, Unterfunktion 1. Vor Aufruf des Interrupts müssen einige Register mit bestimmten Werten belegt werden:

- Register »Regs.CH« mit der Startzeile des Cursorumfangs,
- Register »Regs.CL« mit der Endezeile und
- Register »Regs.AH« mit der Nummer der Unterfunktion 1.

Direktzugriff auf das Betriebssystem mit der Prozedur »MsDos«

Die Prozedur »MsDos« dient ausschließlich dem Aufruf einer Unterfunktion des speziellen Interrupts 21hex. Dieser sogenannte »DOS-Interrupt« sorgt für den direkten Zugang zu den Dienstprogrammen und Funktionen des Betriebssystems. Die Syntax lautet

```
MsDos(<Register>);
```

Dieser Aufruf ist gleichbedeutend mit

```
Intr($21,<Register>);
```

Interrupt 21hex, Funktion 19hex

Mit dieser Funktion wird ermittelt, welches der angeschlossenen Laufwerke das zur Zeit aktuelle Standardlaufwerk ist. Für den Benutzer ist dies jeweils direkt am DOS-Prompt auf dem Bildschirm ersichtlich (beispielsweise »C:\«). Einem Programm, das mit der Laufwerksangabe arbeiten soll, muß diese Information jedoch durch einen Interrupt übermittelt werden.

Eingabe: AH = 19hex
Ausgabe: AL = Code des aktuellen Laufwerks

Anmerkung:
Laufwerk A – in der Regel das erste Diskettenlaufwerk – trägt den Code 0, Laufwerk B den Code 1, Laufwerk C – meist die Festplatte – den Code 2 und so weiter.

Beispiel:
Soll das aktuelle Laufwerk ermittelt werden, so muß zunächst der Interrupt-Funktionswert 19hex in das Prozessorregister AH geladen werden. Dann wird der Interrupt (also das in AH spezifizierte Unterprogramm) aufgerufen. Anschließend kann dem AL-Register der aktuelle Laufwerkscode entnommen werden. Ist dies beispielsweise die 2, so handelt es sich beim momentanen Standardlaufwerk in der Regel um die Festplatte.

Als Beispiel für die Verwendung von »MsDos« soll die bereits erwähnte DOS-Funktion dienen. Sie gibt an der aktuellen Cursorposition ein Zeichen aus. Wieder seien zunächst die Eingabeparameter der Funktion dargestellt. Vor Aufruf des Interrupts müssen geladen werden:

- Register »Regs.AH« mit der Funktionsnummer 2 und
- Register «Regs.DL« mit dem auszugebenden Zeichen.

Das Programm »fillscr.pas« (Listing 2) füllt den Bildschirm mit einem beliebigen Zeichen.

Ergebniswerte aus systemnahen Zugriffen

Die bisher verwendeten Interrupts lieferten keine Informationen zurück, das heißt, nach ihrer Ausführung blieben die Registerwerte unverändert. Dies ist nicht immer der Fall. Die meisten Interrupts liefern nach ihrem Aufruf bestimmte Ergebniswerte zurück. Diese können dann jeweils bestimmten Prozessorregistern entnommen werden. Als Beispiel hierzu dient Listing 3 (»zeit.pas«), das die Systemzeit ermittelt und sie auf den Bildschirm ausgibt. Als Eingabewert benötigen Sie diesmal nur die entsprechende Unterfunktionsnummer des DOS-Interrupts. Dafür müssen nach dem Aufruf gleich

GREENPEACE



Es ist die Menschheit selbst, die Schiffbruch erleidet.

Interrupt 21hex, Funktion 0Ehex

Mit dieser Funktion wird das aktuelle Laufwerk definiert, also dasjenige Laufwerk, auf das bei allen Datenträgeroperationen ohne Pfadangabe automatisch zugegriffen werden soll.

Eingabe: AH = 1Chex
DL = Laufwerksbezeichnung
Ausgabe: AL = Zahl der insgesamt installierten Laufwerke

Anmerkung:
Laufwerk A - in der Regel das 1. Diskettenlaufwerk - trägt den Code 0, Laufwerk B den Code 1, Laufwerk C - meist die Festplatte - den Code 2 und so weiter.
Beispiel:
Soll Laufwerk B als aktuelles Standardlaufwerk definiert werden, so müssen zunächst der Interrupt-Funktionswert 0Ehex in das Prozessorregister AH und der Laufwerkscode 1 in Register DL geladen werden. Dann wird der Interrupt (also das in AH spezifizierte Unterprogramm) aufgerufen. Anschließend kann bei Bedarf aus dem Register AL die Anzahl der insgesamt im System vorhandenen Laufwerke ausgelesen werden.

Interrupt 5

Mit dieser Funktion wird ein Abdruck der aktuellen Textbildschirm-Seite (»Hardcopy«) auf die erste Druckerschnittstelle (»lpt1«) ausgelöst.

Eingabe: keine
Ausgabe: keine

Anmerkung:
Bei dieser BIOS-Funktion handelt es sich um einen der wenigen Hardware-Interrupts.

vier Ausgabewerte verarbeitet werden. Das Registerbild gestaltet sich folgendermaßen:

Eingabe:
- »Regs.AH« wird wie zuvor mit der Funktionsnummer, in diesem Falle 2Chex, belegt.

Ausgabe:
- »Regs.CH« enthält die Stunden.
- »Regs.CL« enthält die Minuten.
- »Regs.DH« enthält die Sekunden.
- »Regs.DL« enthält die 1/100 Sekunden.

Mit diesen Informationen gerüstet, können Sie sich an das Datenträger-Informationsprogramm »Disko« (Listing 4) wagen, mit dessen Hilfe sich komfortabel die folgenden Daten einer Diskette oder Festplatte ermitteln lassen: Sektoren je Cluster, Byte je Sektor, Anzahl der Cluster insgesamt, belegte Speicherkapazität und die noch verbleibende Speicherkapazität.
Hierfür sind in jedem Fall der Interrupt 21hex, die Funktionen 1Chex und 36hex erforderlich. Um das Programm noch etwas interessanter zu gestalten, wurden vier weitere Interrupts verwendet. Zwei Funktionen betreffen den Interrupt 21hex: 19hex zur Ermittlung des Standardlaufwerks sowie 0Ehex zum Setzen des Standardlaufwerks und Ermitteln der Gesamtzahl aller angeschlossenen Laufwerke. Die anderen beiden Funktionen betreffen Interrupt 17hex, Funktion 2 zur Abfrage des Druckerzustands und Interrupt 5 zum Ausdruck des aktuellen Textbildschirms (»Hardcopy«).
Die Syntax des Programmaufrufs lautet

```
Disko <Laufwerk>
```

Die Eingabe »disko a:« würde demnach zur Analyse des Datenträgers in Laufwerk A führen. Fehlt die Laufwerksangabe, so wird der Datenträger im aktuellen Standardlaufwerk untersucht.

(Ralf Martens/bj)

```
[VY3]    1: (* Programm: cursor.pas
[NW6]    2:    Funktion: Verändert die Cursorgroesse
[EK3]    3:    Sprache: Turbo Pascal
[VC3]    4:    Autor: Ralf Martens
[115]    5:    (c)1991 DMV Widuch GmbH & Co. KG *)
[ØØØ]    6:
[MV2]    7: PROGRAM Cursor;
[ØØØ]    8:
[VG1]    9: USES Dos;
[ØØØ]   1Ø:
[XD3]   11: VAR Regs:  Registers;
[5Ø5]   12:            (* zur Aufnahme von Register- *)
[VC3]   13:            (* inhalten                   *)
[ØØØ]   14:
[GY1]   15:     Hoehe: Byte;
[6S4]   16:            (* zur Aufnahme der vom An-   *)
[EN4]   17:            (* wender gewaehlten neuen    *)
[114]   18:            (* Cursorgroesse              *)
[ØØØ]   19:
[MUØ]   2Ø: BEGIN
[XF1]   21:   WriteLn;
[RM5]   22:   Write('Cursor-Höhe = Ø, 1, 2 oder 3 ');
[HR2]   23:   Write('Zeilen? ');
[ØØØ]   24:
[U72]   25:   ReadLn(Hoehe);
[ØØØ]   26:
[SB5]   27:           (* Interrupt-Funktionsnummer:  *)
[LW1]   28:   Regs.AH := 1;
[ØØØ]   29:
[SK4]   3Ø:           (* Cursor-Untergrenze:         *)
[P12]   31:   Regs.CH := 11;
[HS4]   32:           (* Achtung: Wert 11 nur bei    *)
[2Y4]   33:           (* monochromem Monitor; bei    *)
[9V4]   34:           (* Farbe die 11 jeweils durch  *)
[5Y3]   35:           (* eine 6 ersetzen !           *)
[ØØØ]   36:
[1D3]   37:   Regs.CL := (11 + Hoehe);
[ØØØ]   38:
[BM4]   39:           (* Aufruf des Interrupts       *)
[TB2]   4Ø:   Intr($1Ø, Regs);
[ØØØ]   41:
[C25]   42:           (* Falls Cursorgroesse anders  *)
[884]   43:           (* als normal, Meldung:        *)
[472]   44:   IF Regs.CL <> $C
[ZN1]   45:   THEN BEGIN
[YL4]   46:          Write('Rücksetzen durch ');
[Y63]   47:          Write('Aufruf von ');
[HØ5]   48:          Write('''Cursor'' und Eingabe ');
[H52]   49:          Write('von 1 !')
[ZNØ]   5Ø:        END;
[2E1]   51:   WriteLn
[KNØ]   52: END.
```

Listing 1. »cursor.pas« illustriert den Einsatz des Befehls »MsDos«. Mit diesem Programm läßt sich die Cursorgröße manipulieren

```
[8W3]    1: (* Programm: fillscr.pas
[DY5]    2:    Funktion: Füllt den Bildschirm mit
[614]    3:     einem beliebigen Zeichen
[EK3]    4:    Sprache: Turbo Pascal
[VC3]    5:    Autor: Ralf Martens
[115]    6:    (c)1991 DMV Widuch GmbH & Co. KG *)
[ØØØ]    7:
[9X2]    8: PROGRAM FillScr;
[ØØØ]    9:
[X82]   1Ø: USES Crt, Dos;
[ØØØ]   11:
[7V3]   12: VAR Regs:    Registers;
[VC4]   13:              (* zur Aufnahme von Regi-   *)
[BP3]   14:              (* sterinhalten             *)
[ØØØ]   15:
[7U2]   16:     Spalte:  Integer;
[DN2]   17:     Zeile:   Integer;
[ØØØ]   18:
[UB2]   19:     Zeichen: char;
[8L4]   2Ø:              (* zur Aufnahme des vom An- *)
[GC4]   21:              (* wender eingebenden Zei-  *)
[AW2]   22:              (* chens                    *)
[ØØØ]   23:
[MUØ]   24: BEGIN
[XF1]   25:   WriteLn;
[UL6]   26:   Write('Welches Zeichen soll ausgegeben');
[813]   27:   Write(' werden? ');
[3K2]   28:   ReadLn(Zeichen);
[ØØØ]   29:
[2H3]   3Ø:   FOR Spalte := 1 TO 8Ø DO
[BE3]   31:     FOR Zeile := 1 TO 23 DO
[MUØ]   32:       BEGIN
[LM3]   33:         GotoXY(Spalte,Zeile);
[ØØØ]   34:
[DG4]   35:              (* Laden der Funktionsnr.:  *)
[K72]   36:         Regs.AH := $Ø2;
[ØØØ]   37:
[B14]   38:              (* Laden des Zeichens:      *)
[AP3]   39:         Regs.DL := Ord(Zeichen);
[ØØØ]   4Ø:
[KF4]   41:              (* Aufruf des Interrupts:   *)
[8Y1]   42:         MsDos(Regs)
```

```
[4MØ]   43:       END
[KNØ]   44: END.
```

Listing 2. Das Programm »fillscr.pas« dient zur Erläuterung des Intr-Befehls und füllt den gesamten Bildschirm mit einem beliebigen Zeichen

```
[SB3]    1: (* Programm: zeit.pas
[EP5]    2:    Funktion: Ermittelt die aktuelle
[BU4]    3:     Systemzeit und gibt sie aus
[EK3]    4:    Sprache: Turbo Pascal
[VC3]    5:    Autor: Ralf Martens
[115]    6:    (c)1991 DMV Widuch GmbH & Co. KG *)
[ØØØ]    7:
[5A2]    8: PROGRAM Zeit;
[ØØØ]    9:
[VG1]   1Ø: USES Dos;
[ØØØ]   11:
[XD3]   12: VAR Regs:  Registers;
[PZ4]   13:            (* zur Aufnahme von Register-*)
[CC3]   14:            (* inhalten                 *)
[ØØØ]   15:
[5W2]   16:     Stunde:  Integer;
[3V2]   17:     Minute:  Integer;
[EW2]   18:     Sekunde: Integer;
[8Y2]   19:     HundSek: Integer;
[ØØØ]   2Ø:
[MUØ]   21: BEGIN
[DG4]   22:            (* Laden der Funktionsnr.:  *)
[992]   23:   Regs.AH := $2C;
[ØØØ]   24:
[KF4]   25:            (* Aufruf des Interrupts:   *)
[3Ø2]   26:   MsDos(Regs);
[ØØØ]   27:
[2H4]   28:            (* Auslesen der Ergebnis-   *)
[XZ2]   29:            (* werte:                   *)
[V23]   3Ø:   Stunde  := Regs.CH;
[523]   31:   Minute  := Regs.CL;
[M53]   32:   Sekunde := Regs.DH;
[U73]   33:   HundSek := Regs.DL;
[ØØØ]   34:
[XF1]   35:   WriteLn;
[EA5]   36:   Write('Momentane Systemzeit: ');
[FZ5]   37:   Write(Stunde,':',Minute,':',Sekunde);
[DE3]   38:   Write(':',HundSek,'.');
[2E1]   39:   WriteLn
[KNØ]   4Ø: END.
```

Listing 3. »zeit.pas« zeigt die Handhabung von Interrupt-Ausgabewerten anhand der Ermittlung der aktuellen Systemzeit

```
[XF3]    1: (* Programm: disko.pas
[PB6]    2:    Funktion: Ermittlung allgemeiner und
[PN6]    3:     spezieller Datenträgerinformationen
[EK3]    4:    Sprache: Turbo Pascal
[VC3]    5:    Autor: Ralf Martens
[115]    6:    (c)1991 DMV Widuch GmbH & Co. KG *)
[ØØØ]    7:
[XG2]    8: PROGRAM Disko;
[ØØØ]    9:
[9G6]   1Ø: (* Einbinden der Units zur Bildschirm-   *)
[9G6]   11: (* kontrolle und fuer Zugriff auf Be-   *)
[Y26]   12: (* triebssystem-Funktionen:             *)
[ØØØ]   13:
[X82]   14: USES Crt, Dos;
[ØØØ]   15:
[7V3]   16: VAR Regs:    registers;
[VC4]   17:              (* zur Aufnahme von Regi-  *)
[BP3]   18:              (* sterinhalten            *)
[7C2]   19:     Ok:      boolean;
[TM4]   2Ø:              (* Schalter fuer Parameter- *)
[HW2]   21:              (* zeile                   *)
[4X1]   22:     LW:      char;
[2F4]   23:              (* zur Aufnahme des zu ana- *)
[VE4]   24:              (* lysierenden Laufwerks   *)
[3B2]   25:     OrigLW:  char;
[N74]   26:              (* zur Speicherung der an-  *)
[W74]   27:              (* fangs aktuellen Lauf-   *)
[FF3]   28:              (* werksangabe             *)
[GF2]   29:     Antwort: char;
[KV3]   3Ø:              (* fuer  'j/n'-Abfrage     *)
[ØØØ]   31:
[3G4]   32:              (* zur Aufnahme der ermit- *)
[414]   33:              (* telten Disk-Werte:      *)
[ØØØ]   34:
[GY3]   35:     SektorZahl:      integer;
[4E4]   36:     BytesProSektor:  integer;
[V84]   37:     GesamtCluster:   LongInt;
[B24]   38:     GesamtKapaz:     LongInt;
[ØØØ]   39:
[224]   4Ø:     FreiCluster:     LongInt;
[234]   41:     FreiProzent:     LongInt;
[D74]   42:     BelegtCluster:   LongInt;
[MS3]   43:     FreiKapaz:       LongInt;
[ØØØ]   44:
[ØØØ]   45:
[FN4]   46: (* ------------------------------------ *)
[ØØØ]   47:
[Y85]   48: PROCEDURE Fehler(Code: integer);
[ØØØ]   49:
[1A6]   5Ø: (* Funktion: gibt bei falschen Lauf-     *)
[ES6]   51: (* werksangaben eine Fehlermeldung aus   *)
[2T6]   52: (* und bricht die Programmausfuehrung ab *)
[ØØØ]   53:
[ZH2]   54: VAR i: integer;
[L44]   55:       (* Zaehlvariable                   *)
[ØØØ]   56:
[MUØ]   57: BEGIN
[RV3]   58:           (* zeichne Rahmen:            *)
[XF1]   59:   WriteLn;
[4E3]   6Ø:   Write('                  ',#213);
[ØØØ]   61:
[6R2]   62:   FOR i := 1 TO 41 DO
[UN1]   63:     Write(#2Ø5);
[U62]   64:   WriteLn(#184);
[ØØØ]   65:
[CF3]   66:   Write('                  ',#179);
[UR1]   67:   HighVideo;
[ØØØ]   68:
[5Ø5]   69:           (* werte Fehlercode aus und   *)
[WU4]   7Ø:           (* gib entsprechende Fehler-  *)
[7N3]   71:           (* meldung aus:               *)
[FY1]   72:   CASE Code of
[351]   73:     1: BEGIN
[NY3]   74:          Write('       Fehlerhafte ');
[EK4]   75:          Write('Laufwerksangabe!      ')
[ZNØ]   76:        END;
[551]   77:     2: BEGIN
[725]   78:          Write('  Laufwerk ungueltig ');
[EB4]   79:          Write('oder nicht bereit!  ')
[4MØ]   8Ø:        END
[ZNØ]   81:   END;
[ØØØ]   82:
[RV3]   83:           (* zeichne Rahmen:            *)
[9U1]   84:   NormVideo;
[V62]   85:   WriteLn(#179);
[CF3]   86:   Write('                  ',#179);
[N73]   87:   Write('                      ');
[MY3]   88:   WriteLn('                    ',#179);
[CF3]   89:   Write('                  ',#179);
[P86]   9Ø:   Write('  Syntax:    DISKO <gueltiges_');
[BA4]   91:   WriteLn('Laufwerk>: ',#179);
[CF3]   92:   Write('                  ',#179);
[164]   93:   Write('  z.B:       DISKO A:');
[A24]   94:   WriteLn('                    ',#179);
[ØE3]   95:   Write('                  ',#212);
[ØØØ]   96:
[6R2]   97:   FOR i := 1 TO 41 DO
[UN1]   98:     Write(#2Ø5);
[U62]   99:   WriteLn(#19Ø);
[ØØØ]  1ØØ:
[XF1]  1Ø1:   WriteLn;
[RPØ]  1Ø2:   Halt
[ZNØ]  1Ø3: END;
[ØØØ]  1Ø4:
[FN4]  1Ø5: (* ------------------------------------ *)
[ØØØ]  1Ø6:
[EZ2]  1Ø7: PROCEDURE AktuLW;
[ØØØ]  1Ø8:
[117]  1Ø9: (* Funktion: ermittelt das aktuelle, also*)
[NC6]  11Ø: (* das momentane Standard-Laufwerk       *)
[ØØØ]  111:
[3B3]  112: VAR LwNr:    integer;
[FY4]  113:              (* zur Aufnahme des internen *)
[TY3]  114:              (* Laufwerkscodes           *)
[ØØØ]  115:
[MUØ]  116: BEGIN
[UT4]  117:              (* Laden des AH-Registers mit*)
[6V4]  118:              (* Interrupt-Funktions-Nr.   *)
[LV2]  119:              (* 19h:                     *)
[982]  12Ø:   Regs.AH := $19;
[ØØØ]  121:
[ZØ5]  122:              (* Aufruf dieses Interrupts: *)
[3Ø2]  123:   MsDos(Regs);
[ØØØ]  124:
[X15]  125:              (* Auslesen des Inhaltes von *)
[1R4]  126:              (* Register AL, also des in- *)
[4R4]  127:              (* ternen Laufwerkscodes:    *)
[NM2]  128:   LwNr := Regs.AL;
[ØØØ]  129:
[CD4]  13Ø:              (* Umrechnen des Codes in    *)
[YF4]  131:              (* entsprechenden ASCII-     *)
[UH3]  132:              (* Buchstaben:               *)
[YZ2]  133:   LW := Char(LwNr + 65)
[ZNØ]  134: END;
[ØØØ]  135:
[FN4]  136: (* ------------------------------------ *)
[ØØØ]  137:
[6F4]  138: PROCEDURE NeuLw(LW: char);
[ØØØ]  139:
[EV6]  14Ø: (* Funktion: bestimmt ein neues, vom An- *)
[ØW6]  141: (* wender gewaehltes aktuelles Laufwerk  *)
[ØØØ]  142:
[H33]  143: VAR LwNr: integer;
[9Ø5]  144:          (* zur Aufnahme des internen   *)
[LØ4]  145:          (* Laufwerkscodes              *)
[ØØØ]  146:
[MUØ]  147: BEGIN
[WT4]  148:          (* Ermittlung des internen     *)
```

Listing 4. Das Datenträger-Informationsprogramm »disko.pas« gibt zahlreiche Informationen zum gewählten Laufwerk

```
[1K4]  149:            (* Laufwerkscodes aus dem      *)
[RV4]  150:            (* ASCII-Buchstaben des ge-    *)
[DS4]  151:            (* wuenschten Laufwerks:       *)
[L43]  152:    LwNr := Ord(LW) - 65;
[ØØØ]  153:
[MV4]  154:            (* Laden des AH-Registers mit  *)
[U65]  155:            (* Interrupt-Funktions-Nr. ØEh:*)
[B92]  156:    Regs.AH := $ØE;
[ØØØ]  157:
[XX4]  158:            (* Laden des DX-Registers mit  *)
[PØ5]  159:            (* mit dem internen Laufwerks- *)
[PW4]  160:            (* code fuer das gewuenschte   *)
[ZD4]  161:            (* aktuelle Laufwerk:          *)
[WM2]  162:    Regs.DX := LwNr;
[ØØØ]  163:
[T25]  164:            (* Aufruf dieses Interrupts:   *)
[8Y1]  165:    MsDos(Regs)
[ZNØ]  166: END;
[ØØØ]  167:
[FN4]  168: (* ------------------------------------ *)
[ØØØ]  169:
[YX3]  170: FUNCTION Eingabe: char;
[ØØØ]  171:
[MG6]  172: (* Funktion: nimmt die an das Programm   *)
[1N6]  173: (* uebergebenen Parameter entgegen und   *)
[2R4]  174: (* wertet sie aus                        *)
[ØØØ]  175:
[NH3]  176: VAR Temp: string[8Ø];
[HB5]  177:            (* speichert die Parameterzeile*)
[ZØ2]  178:      i:    integer;
[YV3]  179:            (* Zaehlvariable               *)
[ØØØ]  180:
[MUØ]  181: BEGIN
[5D4]  182:            (* Initialisierungen:          *)
[NR2]  183:    FOR i := 1 TO 8Ø DO
[LC2]  184:      Temp[i] := ' ';
[CU1]  185:    Ok := FALSE;
[ØØØ]  186:
[AZ4]  187:            (* Falls nur 1 Parameter ueber-*)
[8Y4]  188:            (* geben, dieser ein Buchstabe *)
[5T4]  189:            (* und an zweiter Stelle ein   *)
[U25]  190:            (* Doppelpunkt, dann setze den *)
[CP4]  191:            (* Schalter 'Ok' auf TRUE =    *)
[X64]  192:            (* alles in Ordnung:           *)
[GM2]  193:    IF ParamCount = 1
[ZN1]  194:    THEN BEGIN
[Ø73]  195:           Temp := ParamStr(1);
[ØØØ]  196:
[C35]  197:           IF (UpCase(Temp[1]) IN ['A'..'Z'])
[AU2]  198:              AND (Temp[2] = ':')
[ZN1]  199:           THEN BEGIN
[ZF4]  200:                  Eingabe := UpCase(Temp[1]);
[EH1]  201:                  Ok := TRUE
[ZNØ]  202:                END;
[ØØØ]  203:
[9V4]  204:            (* Falls die Laenge des Para-  *)
[VT4]  205:            (* meters groesser als 2 Buch- *)
[XR4]  206:            (* staben, setze Schalter 'Ok' *)
[SC3]  207:            (* auf FALSE:                  *)
[Y13]  208:           IF (Length(Temp) > 2)
[3K2]  209:           THEN Ok := FALSE
[ZNØ]  210:         END;
[ØØØ]  211:
[8T4]  212:            (* Falls mehr als 1 Parameter  *)
[415]  213:            (* vorhanden oder der Schalter *)
[1H4]  214:            (* 'Ok' = FALSE, rufe die Feh- *)
[8X4]  215:            (* lermeldungs-Routine auf:    *)
[C65]  216:    IF (ParamCount > 1) OR (Ok = FALSE)
[482]  217:    THEN Fehler(1)
[ZNØ]  218: END;
[ØØØ]  219:
[FN4]  220: (* ------------------------------------ *)
[ØØØ]  221:
[2N4]  222: PROCEDURE DiskTyp(LW: char);
[ØØØ]  223:
[WØ7]  224: (* Funktion: ermittelt Datentraegertyp  *)
[ØØØ]  225:
[H33]  226: VAR LwNr: integer;
[9Ø5]  227:            (* zur Aufnahme des internen   *)
[LØ4]  228:            (* Laufwerkscodes              *)
[ØØØ]  229:
[MUØ]  230: BEGIN
[WT4]  231:            (* Ermittlung des internen     *)
[PU4]  232:            (* werkscodes aus dem eingege- *)
[Y15]  233:            (* benen Laufwerksbuchstaben:  *)
[K43]  234:    LwNr := Ord(LW) - 64;
[ØØØ]  235:
[MV4]  236:            (* Laden des AH-Registers mit  *)
[R65]  237:            (* Interrupt-Funktions-Nr. 1Ch:*)
[792]  238:    Regs.AH := $1C;
[ØØØ]  239:
[DW4]  240:            (* Laden des DL-Registers mit  *)
[ØE4]  241:            (* mit dem internen Code:      *)
[CL2]  242:    Regs.DL := LwNr;
[ØØØ]  243:
[ZN4]  244:            (* Aufruf des Interrupts:      *)
[3Ø2]  245:    MSDos(Regs);
[ØØØ]  246:
[F55]  247:            (* Auslesen der Datentraeger-  *)
[3Ø4]  248:            (* informationen:              *)
[E54]  249:    SektorZahl      := Regs.AL;
```

```
[AP4]  250:    BytesProSektor  := Regs.CX;
[6G4]  251:    GesamtCluster   := Regs.DX;
[ØØØ]  252:
[BC5]  253:            (* Berechnung der theoretischen*)
[755]  254:            (* Gesamtspeicherkapazitaet:   *)
[PN4]  255:    GesamtKapaz     := SektorZahl
[AV2]  256:                       * BytesProSektor
[PP2]  257:                       * GesamtCluster
[ZNØ]  258: END;
[ØØØ]  259:
[FN4]  260: (* ------------------------------------ *)
[ØØØ]  261:
[4F5]  262: PROCEDURE FreiSpeicher(LW: char);
[ØØØ]  263:
[Ø76]  264: (* Funktion: ermittelt den auf dem       *)
[K47]  265: (* Datentraeger noch zur Verfuegung ste- *)
[726]  266: (* henden freien Speicherplatz           *)
[ØØØ]  267:
[H33]  268: VAR LwNr: integer;
[9Ø5]  269:            (* zur Aufnahme des internen   *)
[LØ4]  270:            (* Laufwerkscodes              *)
[ØØØ]  271:
[MUØ]  272: BEGIN
[WT4]  273:            (* Ermittlung des internen     *)
[1W4]  274:            (* Laufwerkscodes aus dem ein- *)
[445]  275:            (* gegebenen Laufwerksbuchst.: *)
[K43]  276:    LwNr := Ord(LW) - 64;
[ØØØ]  277:
[MV4]  278:            (* Laden des AH-Registers mit  *)
[Z55]  279:            (* Interrupt-Funktions-Nr. 36h:*)
[482]  280:    Regs.AH := $36;
[ØØØ]  281:
[DW4]  282:            (* Laden des DL-Registers mit  *)
[554]  283:            (* dem internen Code:          *)
[CL2]  284:    Regs.DL := LwNr;
[ØØØ]  285:
[ZN4]  286:            (* Aufruf des Interrupts:      *)
[3Ø2]  287:    MSDos(Regs);
[ØØØ]  288:
[8A5]  289:            (* Falls Laufwerk nicht bekannt*)
[ØA5]  290:            (* rufe Fehlermeldungs-Routine *)
[NS4]  291:            (* auf, ansonsten lies die Er- *)
[N15]  292:            (* gebniswerte des Interrupts  *)
[TE4]  293:            (* aus den Registern aus:      *)
[YN2]  294:    IF Regs.AX = $FFFF
[ZN1]  295:    THEN BEGIN
[5P1]  296:           NeuLw('C');
[BB1]  297:           Fehler(2)
[4MØ]  298:         END
[WP1]  299:    ELSE BEGIN
[T84]  300:           GesamtCluster  := Regs.DX;
[K34]  301:           FreiCluster    := Regs.BX;
[8Z3]  302:           SektorZahl     := Regs.AX;
[YE4]  303:           BytesProSektor := Regs.CX;
[ØØØ]  304:
[VW4]  305:           (* Berechnung der belegten und *)
[G45]  306:           (* noch zur Verfuegung stehen- *)
[X34]  307:           (* den Kapazitaeten:           *)
[U95]  308:           BelegtCluster := GesamtCluster
[EB2]  309:                            - FreiCluster;
[TD4]  310:           FreiKapaz     := FreiCluster
[822]  311:                            * SektorZahl
[ØZ2]  312:                            * BytesProSektor;
[FV4]  313:           FreiProzent   := (FreiCluster
[3MØ]  314:                            * 1ØØ)
[F43]  315:                            div GesamtCluster
[4MØ]  316:    END
[ZNØ]  317: END;
[ØØØ]  318:
[FN4]  319: (* ------------------------------------ *)
[ØØØ]  320:
[LY2]  321: PROCEDURE Rahmen;
[ØØØ]  322:
[ØL6]  323: (* Funktion: zeichnet einen Rahmen um    *)
[1L5]  324: (* die Ausgabe der Ergebnisse            *)
[ØØØ]  325:
[5V5]  326: (* verwendete ASCII-Zeichen:             *)
[S53]  327: CONST  ObLi       = #2Ø1;
[H62]  328:        UntLi      = #2ØØ;
[372]  329:        UntRe      = #188;
[K22]  330:        ObRe       = #187;
[V62]  331:        Horiz      = #2Ø5;
[P32]  332:        Vert       = #186;
[ØØØ]  333:
[BR2]  334: VAR  i: integer;
[444]  335:         (* Zaehlvariable                  *)
[ØØØ]  336:
[MUØ]  337: BEGIN
[EH4]  338:         (* loesche Bildschirm:            *)
[DB1]  339:    ClrScr;
[ØØØ]  340:
[944]  341:         (* zeichne Linien:                *)
[GV1]  342:    GotoXY(1,1);
[EZ1]  343:    Write(ObLi);
[ØØØ]  344:
[BR2]  345:    FOR i := 2 TO 37 DO
[L82]  346:        Write(Horiz);
[ØØØ]  347:
[DG2]  348:    Write(' DISKO ');
[ØØØ]  349:
[5U2]  350:    FOR i := 45 TO 79 DO
```

Noch erhältlich:

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-Tastatur
- Alle Listings zum Kurs »PC-glasklar«
- Demo für Sprachdigitalisierung

DOS EXTRA Nr. 4

- Desktop-Publishing-Programm »SDTP«
- Kommandoeditor für MS-DOS
- Editor für Disketten und Festplatten
- Speicherresidente Programme auflisten und entfernen
- Paßwort-Schutz für MS-DOS
- Tastatureingabe automatisiert
- Clock-Utility für alle Grafikkarten
- Cursor und Zeichensatz für EGA-Karte
- Fenstertechnik für Turbo C
- Generator für Hilfsbildschirme

DOS EXTRA Nr. 5

- 16 Farben für die CGA-Farbgrafikkarte
- Snapshot: Bildschirmfotos per Software
- Druckerzeichensatz selbstgemacht
- Mausgesteuertes Zeichenprogramm
- Editor in GW-Basic
- Setup-Programm für ATs
- Maschinensprache für GW-Basic
- Erklärtes Listing: das Basic-Spiel »Stardos«

DOS EXTRA Nr. 6

- SBasic-Compiler im Quelltext
- Datenbankprogramm »Parat«
- Fensterorientierte Funktionen für GW-Basic und SBasic
- Turbo-Basic-Bibliothek für 16 Farben
- Analyse von Grafikkarten
- Maskengenerator für Binärdateien
- dBase-Datenbanken in Basic
- Superhirn für Quick Basic 4.0

DOS EXTRA Nr. 7

- MS-DOS-Benutzeroberfläche Easydos
- Benutzeroberfläche für SBasic
- Datengenerator für Basic, C und Pascal
- Sprite-Editor für Farbgrafikkarten
- DOS-Packer
- Virenschutzprogramm
- Weltkarte im Computer
- Einfacher Prozessor-Emulator

DOS EXTRA Nr. 8

- Makroassembler SAM III
- Exe2bin-Ersatz durch "exe2com"
- Programm zur Verbesserung der Speicherverwaltung unter MS-DOS
- Routine zum Speichern von Hardcopies
- Alle Listings zum Einsteigerschwerpunkt und zu den Werkstätten

DOS EXTRA Nr. 9

- Definitive Eingabe für Batchdateien
- Minidatenbank per Batchdatei
- Komfortable MS-DOS-Befehlseingabe
- Utility für sicheres Dateilöschen
- Pulldown-Menüs unter GW-Basic
- Analoguhr für Selfmade-Designer
- Konvertierung vom EGA- in den Hercules-Modus: Schraffur statt Farbe

DOS EXTRA Nr. 10

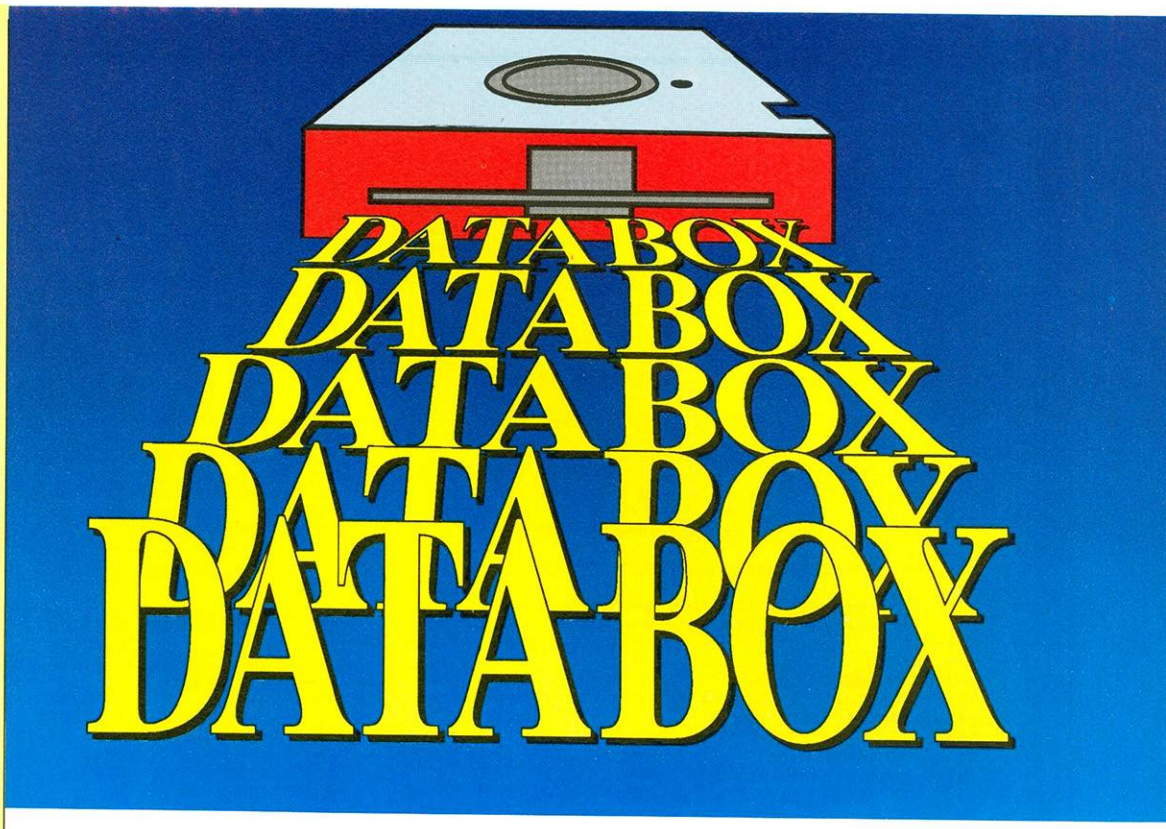
- Units-Bibliotheken für Turbo Pascal: Datum, Sound, Bildschirmausgabe
- Snapshot-Utility, Version 2.7
- TSR-Datei-Manager
- DOS-Assembler SAM III, Version 1.1
- Schnelles Kopierprogramm
- Komfortable Dateiumleitung
- Hercules-Grafik in Basic programmiert
- Patch für BGI-Treiber von Turbo Pascal

DOS EXTRA Nr. 11

- Preiswerte HPGL-Erweiterung, der HPGL-Extender
- Leistungsstarker Disketteneditor
- Unit für die komfortable Speicherverwaltung unter MS-DOS
- Blockbefehle für den Turbo-Pascal-Editor nach dem SAA-Standard
- Sound-Erzeugung unter Turbo Pascal
- Speicherresidente Uhr

DOS EXTRA Nr. 12

- Kreative Grafik in Basic zum Mitmachen
- Subroutinen zur Anpassung von Basic-Programmen an alle PC-Grafikkarten
- Außergewöhnliche Fraktalgrafiken in Turbo Pascal
- Das ultimative Hardcopy-Utility für fast alle Drucker
- Testbild für Monitore in Basic programmiert



Die Software zu den DOS-EXTRA-Heften

Sagenhaft günstig, extra viel gute Software fürs Geld!

DATABOX:

- **mehr als der übliche Softwareservice**
- **bringt alle Programme der DOS EXTRA auf Datenträger**

Für alle PCs unter MS-DOS

Den Aufbau der Databox-Diskette finden Sie in der Datei »read.me« erklärt. Geben Sie deshalb vor dem Start »type read.me« ein. Sämtliche Pascal- und C-Programme liegen auch in einer compilierten Form für Hercules- und Farbgrafikkarten vor. Es ist allerdings nicht gewährleistet, daß diese Versionen auf Hardware, die nicht vollständig kompatibel zum IBM PC ist, problemlos laufen. In diesem Fall müssen Sie den Quellcode nach der Anpassung mit dem entsprechenden Compiler neu übersetzen.
Die Basic-Programme arbeiten mit dem GW-Basic-Interpreter von Microsoft einwandfrei. Bei Interpretern und Compilern anderer Hersteller kann es zu Problemen bei der Programmausführung kommen, sofern diese nicht voll kompatibel sind.

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

35,– DM (unverbindliche Preisempfehlung)
Wenn Sie über den DMV-Bestellservice bestellen, gilt folgendes:

Inland:		**Ausland:**	
Einzelpreis	35,– DM	Einzelpreis	35,– DM
zzgl. Versandkosten	4,– DM	zzgl. Versandkosten	6,– DM
Endpreis	**39,– DM**	**Endpreis**	**41,– DM**

Unabhängig von der Anzahl der bestellten Produkte berechnen wir für das Inland 4,– DM bzw. für das Ausland 6,– DM Porto und Verpackung.

– Bitte benutzen Sie die Bestellkarte. –



DOS EXTRA Nr. 14

- komfortables Datensicherungsprogramm »SBackup« mit Benutzeroberfläche
- professioneller Zeichensatzeditor für 24-Nadeldrucker
- Turbo-Pascal-Unit zur Generierung von 3D-Animationen
- Pascal-Quelltexte in Schreibweise und Wirkung optimieren
- Utilities zur Bearbeitung von Dateien und Verzeichnissen
- Schnelle Grafikroutinen und mathematische Funktionen

DMV-Verlag · Postfach 250 · 3440 Eschwege

DMV Daten- und Medienverlag

```
[L82]  351:       Write(Horiz);
[ØØØ]  352:
[EZ1]  353:   Write(ObRe);
[ØØØ]  354:
[1R2]  355:   FOR i := 2 TO 21 DO
[MUØ]  356:       BEGIN
[112]  357:         GotoXY(1, i);
[U12]  358:         Write(Vert);
[M22]  359:         GotoXY(8Ø,i);
[ZZ1]  36Ø:         Write(Vert)
[ZNØ]  361:       END;
[ØØØ]  362:
[VZ1]  363:   GotoXY(1,22);
[F82]  364:   Write(UntLi);
[ØØØ]  365:
[TR2]  366:   FOR i := 1 TO 78 DO
[L82]  367:       Write(Horiz);
[ØØØ]  368:
[V42]  369:   Write(UntRe)
[ZNØ]  37Ø: END;
[ØØØ]  371:
[FN4]  372: (* ------------------------------------ *)
[ØØØ]  373:
[G23]  374: PROCEDURE Ausgabe;
[ØØØ]  375:
[8D6]  376: (* Funktion: gibt die Ergebnisse der     *)
[8V5]  377: (* Analyse auf dem Bildschirm aus       *)
[ØØØ]  378:
[MUØ]  379: BEGIN
[YZ1]  38Ø:   GotoXY(24,3);
[1R5]  381:   Write('- Informationen zur Disk-');
[783]  382:   Write('Struktur -');
[ØØØ]  383:
[1Ø2]  384:   GotoXY(1Ø,6);
[UB5]  385:   Write('Analysiertes Laufwerk:');
[PZ2]  386:   Write('             ',LW);
[ØØØ]  387:
[DØ2]  388:   GotoXY(1Ø,9);
[M75]  389:   Write('Sektor(en) pro Cluster:');
[HD4]  39Ø:   Write('             ',SektorZahl);
[142]  391:   GotoXY(1Ø,1Ø);
[Y74]  392:   Write('Bytes je Sektor:');
[NX2]  393:   Write('                 ');
[RY3]  394:   Write(BytesProSektor);
[242]  395:   GotoXY(1Ø,11);
[N54]  396:   Write('Cluster insges.:');
[NX2]  397:   Write('                 ');
[ZP3]  398:   Write(GesamtCluster);
[342]  399:   GotoXY(1Ø,12);
[UU6]  4ØØ:   Write('Allg. Gesamtkapazität demnach: ');
[275]  4Ø1:   Write('    ',GesamtKapaz,' Bytes');
[ØØØ]  4Ø2:
[642]  4Ø3:   GotoXY(1Ø,15);
[864]  4Ø4:   Write('Belegte Cluster:');
[NX2]  4Ø5:   Write('                 ');
[HN3]  4Ø6:   Write(BelegtCluster);
[742]  4Ø7:   GotoXY(1Ø,16);
[UT3]  4Ø8:   Write('Freie Cluster:');
[F63]  4Ø9:   Write('                  ');
[8C3]  41Ø:   Write(FreiCluster);
[842]  411:   GotoXY(1Ø,17);
[XR5]  412:   Write('Freie Speicherkapazität ');
[3Ø3]  413:   Write('demnach: ');
[GM4]  414:   Write('  ',FreiKapaz,' Bytes');
[SV4]  415:   Write(' = ca. ',FreiProzent,'%');
[ØØØ]  416:
[DW1]  417:   GotoXY(1,24)
[ZNØ]  418: END;
[ØØØ]  419:
[FN4]  42Ø: (* ------------------------------------ *)
[ØØØ]  421:
[PD3]  422: PROCEDURE Ausdruck;
[ØØØ]  423:
[KA6]  424: (* Funktion: fragt ab, ob der aktuelle   *)
[NM6]  425: (* Bildschirminhalt, also das Ergebnis   *)
[9B6]  426: (* der Analyse, als Hardcopy auf dem     *)
[576]  427: (* Drucker ausgegeben werden soll        *)
[ØØØ]  428:
[MUØ]  429: BEGIN
[842]  43Ø:   GotoXY(21,2Ø);
[2U6]  431:   Write('Ausdruck des Bildschirminhalts ');
[BG2]  432:   Write('(j/n) ? ');
[ØØØ]  433:
[P31]  434:   REPEAT
[A42]  435:     GotoXY(6Ø,2Ø);
[MC2]  436:     Read(Antwort);
[A42]  437:     GotoXY(6Ø,2Ø);
[FM1]  438:     Write(' ');
[9C4]  439:     Antwort := UpCase(Antwort)
[W73]  44Ø:   UNTIL (Antwort = 'J')
[CY2]  441:         OR (Antwort = 'N');
[ØØØ]  442:
[9E2]  443:   IF Antwort = 'J'
[ZN1]  444:   THEN BEGIN
[UZ4]  445:        (* frage, ob Drucker bereit       *)
[N92]  446:          Regs.DX := $ØØ;
[K72]  447:          Regs.AH := $Ø2;
[BC2]  448:          Intr($17,Regs);
[ØØØ]  449:
[C75]  45Ø:        (* Falls Drucker nicht bereit,    *)
[YG5]  451:        (* gib entsprechende Meldung aus  *)
[BX4]  452:        (* und beende das Programm:       *)
[A92]  453:          IF Regs.AH <> $9Ø
[ZN1]  454:          THEN BEGIN
[C42]  455:                 GotoXY(32,21);
[UR1]  456:                 HighVideo;
[EZ3]  457:                 Write ('Druckerfehler!');
[SM1]  458:                 NormVideo
[4MØ]  459:               END
[ØØØ]  46Ø:
[RØ5]  461:         (* Falls Drucker o.k., starte:   *)
[813]  462:          ELSE Intr($Ø5,Regs)
[ZNØ]  463:        END;
[ØØØ]  464:
[DW1]  465:   GotoXY(1,24)
[ZNØ]  466: END;
[ØØØ]  467:
[FN4]  468: (* ------------------------------------ *)
[ØØØ]  469:
[363]  47Ø: PROCEDURE LwRueck;
[ØØØ]  471:
[XE6]  472: (* Funktion: fragt ab, ob eine Rueckkehr *)
[S27]  473: (* zum Ausgangslaufwerk erwuenscht ist;  *)
[WE6]  474: (* falls ja, wird das urspruengliche    *)
[NX6]  475: (* Laufwerk wieder als Standardlaufwerk *)
[CS4]  476: (* eingerichtet                         *)
[ØØØ]  477:
[X94]  478:        (* Zaehlvariablen:               *)
[ZH2]  479: VAR i: integer;
[AV1]  48Ø:     j: integer;
[ØØØ]  481:
[MUØ]  482: BEGIN
[TN5]  483:        (* loesche vorherige Abfragezeile:*)
[842]  484:   GotoXY(21,2Ø);
[7R2]  485:   FOR j := 1 TO 4Ø DO
[FM1]  486:       Write(' ');
[ØØØ]  487:
[ØØØ]  488:
[C42]  489:   GotoXY(23,2Ø);
[2E6]  49Ø:   Write('Zurueck nach Laufwerk ',OrigLw);
[NM2]  491:   Write(' (j/n) ? ');
[ØØØ]  492:
[P31]  493:   REPEAT
[L42]  494:     GotoXY(55,2Ø);
[MC2]  495:     Read(Antwort);
[L42]  496:     GotoXY(55,2Ø);
[FM1]  497:     Write(' ');
[9C4]  498:     Antwort := UpCase(Antwort)
[W73]  499:   UNTIL (Antwort = 'J')
[CY2]  5ØØ:         OR (Antwort = 'N');
[ØØØ]  5Ø1:
[9E2]  5Ø2:   IF Antwort = 'J'
[X73]  5Ø3:   THEN NeuLw(OrigLw);
[ØØØ]  5Ø4:
[DW1]  5Ø5:   GotoXY(1,24)
[ZNØ]  5Ø6: END;
[ØØØ]  5Ø7:
[CK5]  5Ø8: (* ---------- Hauptprogramm ---------- *)
[ØØØ]  5Ø9:
[MUØ]  51Ø: BEGIN
[8D5]  511:        (* ermittle das akt. Laufwerk:   *)
[HC1]  512:   AktuLw;
[ØØØ]  513:
[2C5]  514:        (* speichere das akt. Laufwerk:  *)
[D12]  515:   OrigLw := LW;
[ØØØ]  516:
[A95]  517:        (* falls kein Parameter ueber-   *)
[945]  518:        (* geben, weise das akt. Lauf-   *)
[T95]  519:        (* als zu analysierenden Daten-  *)
[SS3]  52Ø:        (* traeger zu:                   *)
[SS2]  521:   IF ParamCount <> Ø
[ØØØ]  522:
[ZN1]  523:   THEN BEGIN
[Ø42]  524:          LW := Eingabe;
[AG1]  525:          NeuLw(LW)
[ZNØ]  526:        END;
[ØØØ]  527:
[BG5]  528:        (* ermittle Datentraeger-Typ:    *)
[H42]  529:   DiskTyp(LW);
[ØØØ]  53Ø:
[X25]  531:        (* ermittle Kapazitaeten:        *)
[5Y2]  532:   FreiSpeicher(LW);
[ØØØ]  533:
[C44]  534:        (* zeichne Rahmen:               *)
[M91]  535:   Rahmen;
[ØØØ]  536:
[ZD4]  537:        (* gib Ergebnisse aus:           *)
[BD1]  538:   Ausgabe;
[ØØØ]  539:
[T75]  54Ø:        (* frage, ob Hardcopy vom Ergeb- *)
[374]  541:        (* nis-Bildschirm:               *)
[1M1]  542:   Ausdruck;
[ØØØ]  543:
[GL5]  544:        (* falls analysiertes Laufwerk   *)
[9E5]  545:        (* nicht das Ausgangs-Laufwerk,  *)
[C45]  546:        (* frage, ob alter Zustand wie-  *)
[XE5]  547:        (* derhergestellt werden soll:   *)
[SF2]  548:   IF LW <> OrigLw
[442]  549:   THEN LwRueck
[KNØ]  55Ø: END.
[ØØØ]  551:
```

Listing 4. Das Datenträger-Informationsprogramm »disko.pas« gibt zahlreiche Informationen zum gewählten Laufwerk

Turbo Pascal ab Version 5.0

Turbo auch ohne Coprozessor

Nicht jeder hat das Geld, sich einen der begehrten, aber teuren mathematischen Coprozessoren zu kaufen. Will man trotzdem nicht auf Schnelligkeit verzichten, gibt es zwar in Turbo Pascal die Alternative der Emulation, dies bringt jedoch kaum Geschwindigkeitsvorteile.

Die hier vorgestellte Unit »ftrig.pas« (Listing) bringt Ihre mathematischen Programme auch ohne einen teuren Coprozessor auf Trab. Das Programm bietet neue Ersatzroutinen für die Sinus- und Kosinusfunktion an. Weiterhin wurden die Tangens- und Kotangensfunktion implementiert. Diese neuen Routinen arbeiten bis zu fünfmal schneller als die Originalroutinen.

Die Arbeitsweise der Unit geschieht wie folgt:

Im Rechner wird eine Tabelle der Sinuswerte im Intervall $[0;2\pi]$ angelegt. Die Tabelle hätte auch kleiner ausfallen und nur die Sinuswerte im Intervall $[0;1/2\pi]$ erfassen können. Aber der Speicher des PC ist in der Regel groß genug, die Rechenzeit jedoch kostbar. Die Größe und damit auch die Genauigkeit der neuen Funktionen wird durch die Konstante »utile« festgelegt.

Sollte Ihr Programm beispielsweise den Wert »sin (0,5)« benötigen, so schlägt der Rechner in der beschriebenen Tabelle nach und hat das Ergebnis in Mikrosekundenschnelle gefunden. Eine komplizierte Berechnung ist nicht mehr notwendig.

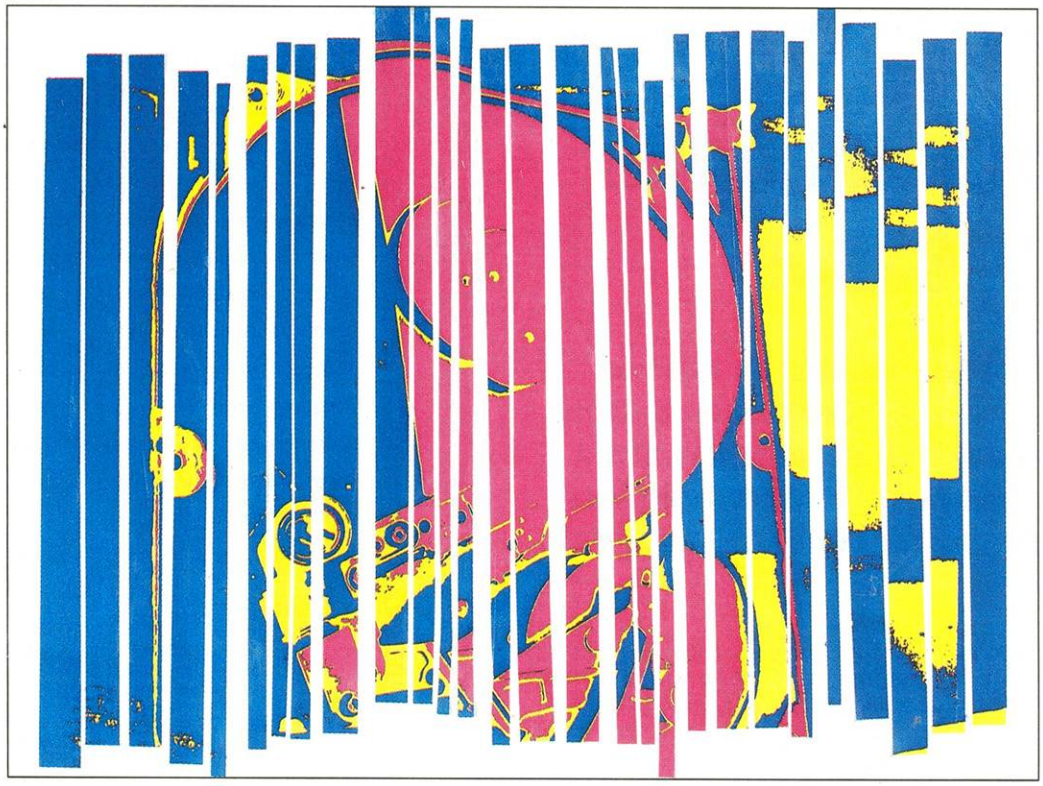
Es wurden für jede Funktion jeweils zwei neue Ersatzfunktionen definiert. Im Fall des Sinus sind dies »BFsin« und »Fsin«. Die Funktion »BFsin« arbeitet in etwa fünfmal schneller, »Fsin« leider »nur« zwei- bis dreimal. Mit der Funktion »Fsin« können auch Werte berechnet werden, die außerhalb des Intervalls $[0;2\pi]$ liegen. Wenn Sie mit der Funktion »BFsin« arbeiten, müssen Sie darauf achten, daß dieses Intervall nicht verlassen wird, da sonst falsche Werte auftreten.

Alle Funktionen nehmen die Einheit »RAD« an.

(Peter Wegner/hs)

```
[5P3]    1: { Programm: ftrig.pas
[EK6]    2:   Funktion: Trigonometrische Funktionen
[EK3]    3:   Sprache: Turbo Pascal
[1C3]    4:   Autor: Peter Wegner
[P35]    5:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    6:
[LØ2]    7: UNIT ftrig;
[ØØØ]    8:
[SK1]    9: INTERFACE
[ØØØ]   1Ø:
[6XØ]   11: CONST
[M52]   12:   uteile  = 1ØØØ;
[ØØØ]   13:
[C57]   14:   { ein größerer Wert ergibt eine höhere
[ØX5]   15:     Genauigkeit. Nachteil: Es wird  mehr
[VB4]   16:     Speicherplatz benötigt }
[ØØØ]   17:
[ØØØ]   18:
[MK2]   19:   Pi      = 3.141592654;
[VH2]   2Ø:   zwei_pi = 2*Pi;
[NH2]   21:   pi_halbe= Pi/2;
[ØE4]   22:   schritt = zwei_pi/uteile;
[ØØØ]   23:
[CG5]   24: VAR s : ARRAY[Ø..uteile] OF real;
[BZ1]   25:     i : INTEGER;
[N91]   26:     j : real;
[ØØØ]   27:
[C45]   28: FUNCTION fsin(x : real) : real;
[ØØØ]   29:
[W25]   3Ø: FUNCTION fcos(x : real) : real;
[ØØØ]   31:
[845]   32: FUNCTION ftan(x : real) : real;
[ØØØ]   33:
[Y25]   34: FUNCTION fcot(x : real) : real;
[ØØØ]   35:
[PY4]   36: FUNCTION bfsin(x : real) : real;
[ØØØ]   37:
[1Z4]   38: FUNCTION bfcos(x : real) : real;
[ØØØ]   39:
[7Y4]   4Ø: FUNCTION bftan(x : real) : real;
[ØØØ]   41:
[4Z4]   42: FUNCTION bfcot(x : real) : real;
[ØØØ]   43:
[4L2]   44: IMPLEMENTATION
[ØØØ]   45:
[C45]   46: FUNCTION fsin(x : real) : real;
[ØØØ]   47:
[MUØ]   48: BEGIN
[HT1]   49:   IF x>=Ø THEN
[MUØ]   5Ø:     BEGIN
[P26]   51:       x:=Abs(x-Trunc(x/zwei_pi)*zwei_pi);
[SC5]   52:       fsin:=s[Abs(Round(x/schritt))]
[4MØ]   53:     END
[ZRØ]   54:   ELSE
[MUØ]   55:     BEGIN
[P26]   56:       x:=Abs(x-Trunc(x/zwei_pi)*zwei_pi);
[3A5]   57:       fsin:=-s[Abs(Round(x/schritt))]
[ZNØ]   58:     END;
[ZNØ]   59: END;
[ØØØ]   6Ø:
[W25]   61: FUNCTION fcos(x : real) : real;
[ØØØ]   62:
[MUØ]   63: BEGIN
[DV3]   64:   fcos:=fsin(x+pi_halbe);
[ZNØ]   65: END;
[ØØØ]   66:
[845]   67: FUNCTION ftan(x : real) : real;
[ØØØ]   68:
[MUØ]   69: BEGIN
[Z73]   7Ø:   ftan:=fsin(x)/fcos(x)
[ZNØ]   71: END;
[ØØØ]   72:
[Y25]   73: FUNCTION fcot(x : real) : real;
[ØØØ]   74:
[MUØ]   75: BEGIN
[TE2]   76:   fcot:=1/ftan(x)
[ZNØ]   77: END;
[ØØØ]   78:
[PY4]   79: FUNCTION bfsin(x : real) : real;
[MUØ]   8Ø: BEGIN
[HT1]   81:   IF x>=Ø THEN
[PC5]   82:     bfsin:=s[Abs(Round(x/schritt))]
[ZRØ]   83:   ELSE
[2E5]   84:     bfsin:=-s[Abs(Round(x/schritt))]
[ZNØ]   85: END;
[ØØØ]   86:
[1Z4]   87: FUNCTION bfcos(x : real) : real;
[ØØØ]   88:
[MUØ]   89: BEGIN
[754]   9Ø:   bfcos:=bfsin(x+pi_halbe);
[ZNØ]   91: END;
[ØØØ]   92:
[7Y4]   93: FUNCTION bftan(x : real) : real;
[ØØØ]   94:
[MUØ]   95: BEGIN
[1Z3]   96:   bftan:=bfsin(x)/bfcos(x)
[ZNØ]   97: END;
[ØØØ]   98:
[4Z4]   99: FUNCTION bfcot(x : real) : real;
[ØØØ]  1ØØ:
[MUØ]  1Ø1: BEGIN
[KL2]  1Ø2:   bfcot:=1/bftan(x)
[ZNØ]  1Ø3: END;
[ØØØ]  1Ø4:
[MUØ]  1Ø5: BEGIN
[WPØ]  1Ø6:   j:=Ø;
[9D3]  1Ø7:   FOR i:=Ø TO uteile DO
[MUØ]  1Ø8:     BEGIN
[L62]  1Ø9:       s[i]:=Sin(j);
[W52]  11Ø:       j:=j+schritt;
[ZNØ]  111:     END;
[KNØ]  112: END.
```

Listing. Die Unit »ftrig.pas« stellt Ihnen nach dem Einbinden in Ihre eigenen Programme sehr schnelle trigonometrische Funktionen zur Verfügung



Turbo Pascal ab Version 5.0

F-Tools: Kampf dem Festplattenchaos

Wer bei der täglichen Arbeit mit MS-DOS bisher wichtige Tools zum Datei-Handling vermißt hat, dem kann jetzt geholfen werden. Fünf kleine und einfache »Helferlein« in Turbo Pascal erleichtern den Umgang mit Laufwerken, Dateien und Verzeichnissen erheblich.

Das hierarchische Dateisystem unter MS-DOS bietet neben großartigen Fähigkeiten auch etliche Probleme, die vor allem Einsteigern die Arbeit mehr als notwendig erschweren kann. Hat man sich mit Müh' und Not dazu durchgerungen, seine Daten in unterschiedlichen Verzeichnissen abzulegen, wird man bald feststellen, daß mit unter Schwierigkeiten entstehen, wenn man auf diese Daten zugreifen möchte.

- Wo war doch gleich eine Datei gespeichert? Bei einem stark verzweigten Verzeichnisdschungel ist es nicht ganz einfach, sich dorthin zurückzubegeben.
- Sind Dateien unter verschiedenen Pfaden mehrfach abgespeichert, und wenn ja – wo?
- Sie möchten sich von einem kompletten Verzeichniszweig (eventuell mit Unterverzeichnissen) trennen. Dazu beginnen Sie mit der Befehlsfolge:

```
cd <Verzeichnis>
del *.*
cd ..
rd <Verzeichnis>
```

Das geht auch einfacher.

Um diese Probleme zu lösen, sind die folgenden kleinen Tools in Turbo Pascal entstanden. In fast allen werden dabei die Verzeichnisinformationen in eine verkettete Liste eingelesen. Diese stehen dann zur weiteren Bearbeitung zur Verfügung. Jedes der Programme verfügt über eine Prozedur mit dem Namen »ftw« (»FileTreeWalk«), in der rekursiv das gesamte Laufwerk (in der Regel Laufwerk C) im Sinne der jeweiligen Problemstellung bearbeitet wird. Vom Grundaufbau her ist diese Funktion überall gleich. Sie unterscheidet sich lediglich in den Aktionen, welche auf die gefundenen Verzeichnisse und/oder Dateien angewendet werden.

Je nach Geschwindigkeit der Festplatte sowie dem Umfang und der Anzahl der zu durchsuchenden Verzeichnisse kann es einige Sekunden dauern, bis man eine Ausgabe erhält. Alle Programme können mit Hilfe einer Umleitung der Ausgabe in eine Datei oder auf einen Drucker dazu gebracht werden, den Anwender (etwa im Rahmen eines Stapel-Jobs) von Bildschirmausgaben zu verschonen.

■ »ff« – Finde Dateien

Oft ist man in der Verlegenheit wissen zu wollen, ob und wenn ja, wo eine bestimmte Datei (oder auch eine Gruppe von Dateien) gespeichert ist. Diese Information wird auf sehr einfache Weise durch das Programm »ff.pas« (Listing 1) geliefert.

Geben Sie dem Programm den Dateinamen, der gesucht werden soll, als Aufrufargument mit. Dabei sind Stellvertreterzeichen (»*« und »?«) erlaubt und werden genauso behandelt wie etwa beim Standardbefehl »dir«.

```
ff *.f
```

findet alle Dateien mit der Namenserweiterung ».f« in allen Verzeichnissen.

```
ff a??.pas
```

findet alle Dateien, deren Namen aus drei Zeichen bestehen, mit »a« beginnen und die Erweiterung »pas« haben, in allen Verzeichnissen.

```
ff *.*
```

findet alle Dateien in allen Verzeichnissen.

```
ff
```

(ohne Argument) arbeitet genauso wie

```
ff *.*
```

Konnte »ff« keine Datei (oder kein Verzeichnis) finden, auf die die übergebene Suchmaske zutrifft, meldet sich »ff« mit der lapidaren Bemerkung »Sorry !!!«.

■ »fdir« – Finde Verzeichnis

Dieses Programm steht in enger Beziehung zu »ff«. Angenommen, Sie möchten die Datei »test.xyz« weiterbearbeiten. Sie haben sich nicht gemerkt, wo diese Datei gespeichert ist. Mit »ff« bekommen Sie die Information, daß sich diese Datei in dem Unterverzeichnis »c:\tp\progssss\nonsense« befindet.

»fdir.pas« (Listing 2) erlaubt Ihnen, ein bestimmtes Verzeichnis zu betreten, indem nur der letzte Teil des Pfadnamens als Parameter angegeben wird. Für das obige Beispiel muß der Aufruf demnach lauten:

```
fdir nonsense
```

»fdir« sucht jetzt die Festplatte nach dem ersten Auftreten eines Unterverzeichnisses »nonsense« ab. Dieses wird dann zum aktuellen Arbeitsverzeichnis gemacht.

Existieren mehrere Unterverzeichnisse mit diesem Namen, läßt sich ein bestimmtes suchen.

```
fdir nonsense 3
```

versucht das dritte Unterverzeichnis »nonsense« auf der Festplatte zu finden. Ausschlaggebend ist dabei die Reihenfolge, in der diese aufgezeichnet wurden. Die gefundenen Verzeichnisse werden dann zum Beispiel in der Form »c:\tp\prog\nonsense« am Bildschirm angezeigt. War das Bemühen erfolglos, das gesuchte Verzeichnis auf der Festplatte zu finden, wird dies mit »fdir: Sorry« mitgeteilt. Im Erfolgsfall erhalten Sie durch »fdir: ok« die Bestätigung für den Verzeichniswechsel.

■ »fdd« - Finde doppelte Verzeichnisse

Sehr oft ist die Festplatte derart mit Unterverzeichnissen belegt, daß ein Verzeichnisname mehrfach vorkommt. Um sich davon ein Bild machen zu können, wird das Programm »fdd.pas« (Listing 3) eingesetzt. Dieses stellt fest, wo überall beispielsweise das Verzeichnis »nonsense« vorhanden ist. Nach dem Starten mit

```
fdd c:
```

liest das Programm die Verzeichnisstruktur der gesamten Festplatte (hier C) und gibt alle mehrfach gefundenen Pfade (zusammen mit dem Datum der Generierung) auf den Bildschirm aus. Wird die Laufwerksangabe unterlassen, greift »fdd« auf das momentane Laufwerk zu.

■ »fdf« – Finde doppelte Dateien

Dieses kleine Utility kann wesentlich wichtiger sein als die Fähigkeiten, die mit »fdd« geboten werden. Nach einiger Zeit sammeln sich auf einer Festplatte die Dateien, man hat ja genügend Platz dazu. Oft geschieht es, daß Dateien in verschiedenen Verzeichnissen gehalten werden. Auf Dauer verlieren Sie so die Übersicht, ob auch wirklich noch alle Versionen notwendig sind.

Einen genauen Überblick über die derzeitige Situation verschafft Ihnen »fdf.pas« (Listing 4). Es zeigt den Namen, den Pfad, die Größe (in Byte) sowie Datum und Uhrzeit der letzten Veränderung für alle mehrfach vorhandenen Dateien an. Beim Aufruf des Programms mit

```
fdf d:
```

wird das als Parameter übergebene Laufwerk (in diesem Fall D) durchsucht. Wird dieser weggelassen, nimmt »fdf« das aktuelle Laufwerk an.

■ »fdel« – Dateien und Verzeichnisse löschen

Wer hat sich nicht schon geärgert, wenn er ein umfangreiches Programmpaket mit mehreren Unterverzeichnissen wieder von seiner Festplatte löschen wollte. Hier darf man zeigen, daß man seine DOS-Befehle intus hat, und auch einige Tipparbeit keine Mühe bereitet.

Wenn Ihnen dies bisher zuviel war, hilft Ihnen ab sofort das Programm »fdel.pas« (Listing 5).

Dabei sollten Sie jedoch mit äußerster Vorsicht vorgehen, um nicht ungewollt wichtige Daten zu löschen. Um dies zu verhindern, starten Sie das Programm mit

```
fdel nonsense /?
```

»fdel« beginnt nun, das angegebene Verzeichnis, in diesem Beispiel »nonsense«, zu durchsuchen, und zeigt Ihnen jede Datei am Bildschirm. Dabei fragt es, ob Sie löschen wollen oder nicht. Das besondere an »fdel« ist jedoch, daß es mit dem Verzeichnis »nonsense« nicht aufhört, sondern auch noch in den darunterliegenden Verzeichnissen weitermacht.

Lassen Sie beim Aufruf den Parameter »/?« weg, löscht »fdel« ohne Abfrage. Dies ist zwar einfacher und geht schneller, kann jedoch bei unvorsichtiger Anwendung zu Datenverlusten führen. (Georg Pohl/hs)

```
[E23]    1: { Programm: ff.pas
[X34]    2:   Funktion: Dateien suchen
[YY5]    3:   Sprache:  Turbo Pascal ab Version 5.Ø
[W63]    4:   Autor:    Georg Pohl
[P35]    5:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    6:
[VG1]    7: USES Dos;
[ØØØ]    8:
[STØ]    9: TYPE
[1P3]   1Ø:   filestr  = STRING[13];
[NH3]   11:   drivestr = STRING[3];
[283]   12:   listeptr = ^ liste;
[FE2]   13:   liste    = RECORD
[D73]   14:                filename : pathstr;
[S23]   15:                next     : listeptr;
[ZNØ]   16:              END;
[6XØ]   17: CONST
[EF2]   18:   backslash  = '\';
[V82]   19:   maske      = '*.*';
[4G2]   2Ø:   nullstring = '';
[YK6]   21:   startmsg   = 'Durchsuche Verzeichnisse ';
[TN4]   22:   copyright  = 'FF - Find Files ';
[GB3]   23:   sorry      = 'Sorry !!!';
[4Z3]   24: VAR tosearch : filestr;
[G13]   25:     tree     : listeptr;
[ØØØ]   26:
[ZP6]   27: FUNCTION isdir(name : pathstr) : BOOLEAN;
[MUØ]   28: BEGIN
[2D3]   29:   isdir:= name[1] = '.';
[ZNØ]   3Ø: END;
```

```
[ØØØ]   31:
[6E6]   32: FUNCTION getcurrentdrive : drivestr;
[683]   33: VAR home : pathstr;
[MUØ]   34: BEGIN
[SD2]   35:   GetDir(Ø, home);
[MR5]   36:   getcurrentdrive:= Copy(home, 1, 2);
[ZNØ]   37: END;
[ØØØ]   38:
[1A5]   39: PROCEDURE push(VAR l : listeptr;
[LZ2]   4Ø:                newval : pathstr);
[WF2]   41: VAR neueliste,
[3N3]   42:     suchliste : listeptr;
[MUØ]   43: BEGIN
[RR2]   44:   New(neueliste);
[H55]   45:   neueliste^. filename:= newval;
[YA4]   46:   neueliste^.next     := NIL;
[5H2]   47:   IF (l = NIL) THEN
[V72]   48:     l:= neueliste
[ZRØ]   49:   ELSE
[MUØ]   5Ø:     BEGIN
[1D2]   51:       suchliste:= l;
[916]   52:       WHILE (suchliste^.filename <> newval)
[7E5]   53:         AND (suchliste^.next <> NIL) DO
[B15]   54:         suchliste:= suchliste^.next;
[AL5]   55:       IF (suchliste^.filename <> newval)
[2RØ]   56:         THEN
[CØ5]   57:         suchliste^.next:= neueliste;
```

Listing 1. Mit »ff.pas« hat das Suchen einer bestimmten Datei auf der Festplatte ein Ende

```
[ZNØ]   58:     END;
[ZNØ]   59: END;
[ØØØ]   6Ø:
[YZ4]   61: PROCEDURE ftw(path : pathstr);
[YC1]   62: VAR buf,
[TN2]   63:     buf2 : searchrec;
[MUØ]   64: BEGIN
[UN5]   65:   FindFirst(path + backslash + maske,
[FB2]   66:              anyfile, buf);
[5P3]   67:   WHILE (doserror = Ø) DO
[MUØ]   68:     BEGIN
[4L4]   69:       IF (NOT isdir(buf.name)) THEN
[MUØ]   7Ø:         BEGIN
[3W4]   71:           IF (buf.attr = directory) THEN
[MUØ]   72:             BEGIN
[7H4]   73:               FindFirst(path + backslash +
[2L1]   74:                         buf.name +
[VS1]   75:                         backslash +
[KY2]   76:                         tosearch,anyfile,
[GYØ]   77:                         buf2);
[5P3]   78:               WHILE (doserror = Ø) DO
[MUØ]   79:                 BEGIN
[2W3]   8Ø:                   IF (NOT isdir(buf2.name))
[2RØ]   81:                     THEN
[6P2]   82:                     push(tree,path +
[VS1]   83:                          backslash +
[2L1]   84:                          buf.name +
[VS1]   85:                          backslash +
[NU1]   86:                          buf2.name);
[XL2]   87:                     FindNext(buf2);
[ZNØ]   88:                 END;
[Z32]   89:               doserror:= Ø;
[4G3]   9Ø:               ftw(path + backslash +
[LF1]   91:                 buf.name)
[ZNØ]   92:             END;
[ZNØ]   93:         END;
[ND2]   94:       FindNext(buf);
[ZNØ]   95:     END;
[ZNØ]   96: END;
[ØØØ]   97:
[TG3]   98: PROCEDURE showtree;
[5Z1]   99: VAR olddir,
[4X2]  1ØØ:     thisdir : dirstr;
[VH2]  1Ø1:     ext     : extstr;
[3P2]  1Ø2:     name    : namestr;
[MUØ]  1Ø3: BEGIN
[LK3]  1Ø4:   olddir:= nullstring;
[4Z2]  1Ø5:   IF (tree = NIL) THEN
[4K2]  1Ø6:     WriteLn(sorry)
[ZRØ]  1Ø7:   ELSE
[A93]  1Ø8:     WHILE (tree <> NIL) DO
[MUØ]  1Ø9:       BEGIN
[ZØ6]  11Ø:         FSplit(tree^.filename,thisdir,name,
[1WØ]  111:                ext);
[P84]  112:         IF (thisdir <> olddir) THEN
[MUØ]  113:           BEGIN
[B13]  114:             WriteLn(thisdir);
[6W2]  115:             olddir:= thisdir;
[ZNØ]  116:           END;
[3L4]  117:         WriteLn(nullstring:5,name,
[VF4]  118:                 nullstring:8-Length(name),
[1WØ]  119:                 ext);
[6Z2]  12Ø:       tree:= tree^.next;
[ZNØ]  121:     END;
[ZNØ]  122: END;
[ØØØ]  123:
[MUØ]  124: BEGIN
[HW3]  125:   tosearch:= ParamStr(1);
[S25]  126:   IF (tosearch = nullstring) THEN
[CU2]  127:     tosearch:= maske;
[9R1]  128:   New(tree);
[KU1]  129:   tree:= NIL;
[8F3]  13Ø:   WriteLn(copyright);
[783]  131:   WriteLn(startmsg);
[AV3]  132:   ftw(getcurrentdrive);
[YL1]  133:   showtree;
[KNØ]  134: END.
```

Listing 1. Mit »ff.pas« hat das Suchen einer bestimmten Datei auf der Festplatte ein Ende

```
[YE3]    1: { Programm: fdir.pas
[666]    2:   Funktion: in Verzeichnisse irgendwo
[4E1]    3:             wechseln
[YY5]    4:   Sprache:  Turbo Pascal ab Version 5.Ø
[W63]    5:   Autor:    Georg Pohl
[P35]    6:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    7:
[ØL1]    8: USES  Dos;
[ØØØ]    9:
[6XØ]   1Ø: CONST
[WD2]   11:   backslash = '\';
[ZG2]   12:   maske     = '\*.*';
[1D5]   13:   usage     = 'Aufruf: FDIR Ziel-Dir'+
[XL2]   14:               ' [Vorkommen]';
[BY4]   15: VAR   gesucht     : STRING[13];
[K13]   16:       found       : BOOLEAN;
[VF3]   17:       homepath    : pathstr;
[SN1]   18:       occurance,
[FS1]   19:       founddirs,
[VY2]   2Ø:       err         : INTEGER;
```

```
[ØØØ]   21:
[YZ4]   22: PROCEDURE ftw(path : pathstr);
[XB3]   23: VAR buf : searchrec;
[MUØ]   24: BEGIN
[E43]   25:   IF (NOT found) THEN
[MUØ]   26:     BEGIN
[5V5]   27:       IF (path[Length(path)] = backslash)
[2RØ]   28:         THEN
[UP4]   29:         Delete(path, Length(path), 1);
[UM5]   3Ø:         FindFirst(path + maske, directory,
[UTØ]   31:                   buf);
[ZNØ]   32:     END;
[Z56]   33:   WHILE (doserror = Ø) AND (NOT found) DO
[MUØ]   34:     BEGIN
[U64]   35:       IF (buf.name[1] <> '.') THEN
[MUØ]   36:         BEGIN
[3W4]   37:           IF (buf.attr = directory) THEN
[MUØ]   38:             BEGIN
[RD4]   39:               IF (buf.name = gesucht) THEN
[MUØ]   4Ø:                 BEGIN
[6H2]   41:                   Dec(occurance);
[4L3]   42:                   found := occurance = Ø;
[E43]   43:                   IF (NOT found) THEN
[MUØ]   44:                     BEGIN
[2N2]   45:                       Inc(founddirs);
[T73]   46:                       WriteLn(founddirs,
[PF1]   47:                               '.  ',path +
[VS1]   48:                                backslash +
[4N1]   49:                               buf.name);
[4MØ]   5Ø:                     END
[ZRØ]   51:                   ELSE
[WH3]   52:                     ChDir(path+backslash +
[4N1]   53:                           buf.name);
[4MØ]   54:                 END
[ZRØ]   55:               ELSE
[LC3]   56:                 ftw(path+backslash +
[LF1]   57:                   buf.name)
[ZNØ]   58:             END;
[ZNØ]   59:         END;
[ND2]   6Ø:         FindNext(buf);
[ZNØ]   61:     END;
[ZNØ]   62: END;
[ØØØ]   63:
[AM6]   64: FUNCTION upcasestr(s : STRING) : STRING;
[WE3]   65: VAR stelle : INTEGER;
[MUØ]   66: BEGIN
[CY4]   67:   FOR stelle := 1 TO Length(s) DO
[1Y2]   68:     CASE s[stelle] OF
[VV3]   69:       'ä' : s[stelle] := 'Ä';
[KX3]   7Ø:       'ö' : s[stelle] := 'Ö';
[RX3]   71:       'ü' : s[stelle] := 'Ü';
[ZRØ]   72:     ELSE
[DK5]   73:       s[stelle] := UpCase(s[stelle]);
[ZNØ]   74:     END;
[AK2]   75:   upcasestr := s;
[ZNØ]   76: END;
[ØØØ]   77:
[MUØ]   78: BEGIN
[RN3]   79:   IF (ParamCount = Ø) THEN
[MUØ]   8Ø:     BEGIN
[9P2]   81:       WriteLn(usage);
[T41]   82:       Halt(1);
[ZNØ]   83:     END;
[CE2]   84:   founddirs := Ø;
[SB2]   85:   occurance := 1;
[VN3]   86:   IF (ParamCount > 1) THEN
[MUØ]   87:     BEGIN
[SC5]   88:       Val(ParamStr(2), occurance, err);
[PA5]   89:       IF (err > Ø) OR (occurance < 1) THEN
[MUØ]   9Ø:         BEGIN
[9P2]   91:           WriteLn(usage);
[T41]   92:           Halt(1);
[ZNØ]   93:         END;
[ZNØ]   94:     END;
[WK5]   95:   gesucht := upcasestr(ParamStr(1));
[T53]   96:   GetDir(Ø, homepath);
[A35]   97:   homepath := Copy(homepath, 1, 3);
[BF2]   98:   found := FALSE;
[HS6]   99:   WriteLn('Suche ',occurance,'. Vorkommen',
[ZL2]  1ØØ:           ' von ', gesucht);
[ØG2]  1Ø1:   ftw(homepath);
[E43]  1Ø2:   IF (NOT found) THEN
[6M3]  1Ø3:     WriteLn('FDIR: Sorry')
[ZRØ]  1Ø4:   ELSE
[MUØ]  1Ø5:     BEGIN
[T53]  1Ø6:       GetDir(Ø, homepath);
[6D5]  1Ø7:       WriteLn('FDIR: ok (', homepath, ')');
[ZNØ]  1Ø8:     END;
[KNØ]  1Ø9: END.
```

Listing 2. Auch Verzeichnisse lassen sich mit »fdir.pas« einfach und schnell lokalisieren

```
[KD3]    1: { Programm: fdd.pas
[4Z5]    2:   Funktion: Verzeichnischaos lüften
[YY5]    3:   Sprache:  Turbo Pascal ab Version 5.Ø
[W63]    4:   Autor:    Georg Pohl
[P35]    5:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    6:
[VG1]    7: USES Dos;
```

```
[ØØØ]    8:
[6XØ]    9: CONST
[N42]   1Ø:   mask      = '*.*';
[WD2]   11:   backslash = '\';
[774]   12:   startmsg  = 'Durchsuche' +
[9E3]   13:               ' Verzeichnisse ... ';
[Ø52]   14:   maxparm   = 1;
[ØØØ]   15:
[RØ5]   16: TYPE   filestr    = STRING[13];
[EF3]   17:        str2       = STRING[2];
[Ø24]   18:        doubles    = ^duplicates;
[R93]   19:        duplicates = RECORD
[5S2]   2Ø:                       path    : pathstr;
[EP2]   21:                       next    : doubles;
[ZNØ]   22:                     END;
[FE3]   23:        dirs       = ^dirstack;
[E23]   24:        dirstack   = RECORD
[1Y2]   25:                       name      : filestr;
[3Ø3]   26:                       path      : pathstr;
[HD3]   27:                       duplicate : doubles;
[VB2]   28:                       next      : dirs;
[ZNØ]   29:                     END;
[ØØØ]   3Ø:
[TA3]   31: VAR    dirst    : dirs;
[UM2]   32:        hd_drive : str2;
[ØØØ]   33:
[CM6]   34: PROCEDURE checkdirname(VAR liste: dirs;
[XG2]   35:                            path    : dirstr;
[ZU2]   36:                            name    : filestr);
[FE2]   37: VAR newfddrec,
[5U2]   38:     actualrec : dirs;
[LA3]   39:     duplrec   : doubles;
[MUØ]   4Ø: BEGIN
[8W3]   41:   path:= path + backslash;
[AP2]   42:   New(newfddrec);
[1E4]   43:   newfddrec^.name      := name;
[7G4]   44:   newfddrec^.path      := path;
[ZU4]   45:   newfddrec^.duplicate := NIL;
[UD4]   46:   newfddrec^.next      := NIL;
[Ø92]   47:   New(duplrec);
[R64]   48:   duplrec^.path        := path;
[E34]   49:   duplrec^.next        := NIL;
[A93]   5Ø:   IF (liste = NIL) THEN
[MUØ]   51:     BEGIN
[443]   52:       liste:= newfddrec;
[ØWØ]   53:       Exit;
[ZNØ]   54:     END;
[DZ2]   55:   actualrec:=liste;
[1C5]   56:   WHILE (actualrec^.next <> NIL) AND
[8H4]   57:         (actualrec^.name <> name) DO
[HY4]   58:     actualrec:= actualrec^.next;
[N85]   59:   IF (actualrec^.name <> name) THEN
[FU4]   6Ø:     actualrec^.next:= newfddrec
[ZRØ]   61:   ELSE
[MUØ]   62:     BEGIN
[ZS5]   63:       WHILE (actualrec^.duplicate <> NIL)
[E84]   64:         DO actualrec^.duplicate:=
[BY4]   65:            actualrec^.duplicate^.next;
[UD5]   66:       actualrec^.duplicate:= duplrec;
[ZNØ]   67:     END;
[ZNØ]   68: END;
[ØØØ]   69:
[YZ4]   7Ø: PROCEDURE ftw(path : pathstr);
[XB3]   71: VAR buf : searchrec;
[MUØ]   72: BEGIN
[UØ6]   73:   FindFirst(path + '\*.*', anyfile, buf);
[5P3]   74:   WHILE (doserror = Ø) DO
[MUØ]   75:     BEGIN
[U64]   76:       IF (buf.name[1] <> '.') THEN
[MUØ]   77:         BEGIN
[3W4]   78:           IF (buf.attr = directory) THEN
[MUØ]   79:             BEGIN
[W54]   8Ø:               checkdirname(dirst, path,
[4N1]   81:                            buf.name);
[YR4]   82:               ftw(path+backslash+buf.name)
[ZNØ]   83:             END;
[ZNØ]   84:         END;
[ND2]   85:       FindNext(buf);
[ZNØ]   86:     END;
[ZNØ]   87: END;
[ØØØ]   88:
[AB7]   89: PROCEDURE showduplicates(VAR liste : dirs);
[ZE3]   9Ø: VAR duprec : doubles;
[ØP2]   91:     fsize  : LONGINT;
[AR2]   92:     fdate  : datetime;
[ØØØ]   93:
[ND5]   94: FUNCTION fill(w, z : WORD) : str2;
[EX2]   95: VAR tmp : STRING;
[LØ2]   96:     p   : INTEGER;
[MUØ]   97: BEGIN
[5E2]   98:   Str(w:z, tmp);
[WM2]   99:   p:= Pos(' ', tmp);
[B42]  1ØØ:   WHILE (p > Ø) DO
[MUØ]  1Ø1:     BEGIN
[SØ2]  1Ø2:       tmp[p]:= 'Ø';
[WM2]  1Ø3:       p:= Pos(' ', tmp);
[ZNØ]  1Ø4:     END;
[SU1]  1Ø5:   fill:= tmp;
[ZNØ]  1Ø6: END;
[ØØØ]  1Ø7:
[ZS6]  1Ø8: PROCEDURE fileinfos(path, name : STRING;
```

```
[XK3]  1Ø9:                          VAR date   : datetime);
[3B3]  11Ø: VAR  buf : searchrec;
[MUØ]  111: BEGIN
[MV5]  112:   FindFirst(path + name, anyfile, buf);
[ZC3]  113:   IF (doserror = Ø) THEN
[PF4]  114:     UnpackTime(buf.time, date);
[ZNØ]  115: END;
[ØØØ]  116:
[PE6]  117: PROCEDURE writeinfos(date : datetime);
[MUØ]  118: BEGIN
[WB2]  119:   Write(' vom ');
[EY1]  12Ø:   WITH date DO
[MUØ]  121:     BEGIN
[6E5]  122:       WriteLn(day,'.',month,'.',year,',
[HN2]  123:               ,fill(hour,2), ':',
[Z44]  124:               fill(min, 2),':',fill(sec,2),
[4GØ]  125:               ')');
[ZNØ]  126:     END;
[ZNØ]  127: END;
[ØØØ]  128:
[MUØ]  129: BEGIN
[P31]  13Ø:   REPEAT
[9E5]  131:     IF (liste^.duplicate <> NIL) THEN
[MUØ]  132:       BEGIN
[MU5]  133:         fileinfos(liste^.path,liste^.name,
[151]  134:                   fdate);
[LK4]  135:         WriteLn(liste^.name, ' in:');
[ØY3]  136:         Write('':5, liste^.path);
[Z23]  137:         writeinfos(fdate);
[AK4]  138:         duprec:= liste^.duplicate;
[AN3]  139:         WHILE (duprec <> NIL) DO
[MUØ]  14Ø:           BEGIN
[614]  141:             fileinfos(duprec^.path,
[Z83]  142:                       liste^.name, fdate);
[M64]  143:             Write('':5, duprec^.path);
[Z23]  144:             writeinfos(fdate);
[9T3]  145:             duprec:= duprec^.next;
[ZNØ]  146:           END;
[ZNØ]  147:       END;
[TH3]  148:     liste:= liste^.next;
[B23]  149:   UNTIL (liste = NIL);
[ZNØ]  15Ø: END;
[ØØØ]  151:
[MUØ]  152: BEGIN
[WZ1]  153:   New(dirst);
[975]  154:   hd_drive := ParamStr(maxparm);
[A22]  155:   dirst:= NIL;
[783]  156:   WriteLn(startmsg);
[7K2]  157:   ftw(hd_drive);
[9V3]  158:   showduplicates(dirst);
[KNØ]  159: END.
```

Listing 3. »fdd.pas« bringt Licht ins Chaos der Unterverzeichnisse

```
[UD3]    1: { Programm: fdf.pas
[H76]    2:   Funktion: mehrfach vorhandene Dateien
[D31]    3:             suchen
[YY5]    4:   Sprache:  Turbo Pascal ab Version 5.Ø
[W63]    5:   Autor:    Georg Pohl
[P35]    6:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    7:
[P52]    8: {$F+,I+,V-,R-}
[VG1]    9: USES Dos;
[ØØØ]   1Ø:
[6XØ]   11: CONST
[VS5]   12:   mem_err   = 'Nicht genug Speicher! '+
[DK3]   13:               '- Programm bricht ab!';
[RS5]   14:   open_err  = ', kann nicht geöffnet'+
[MP1]   15:               ' werden)';
[CP6]   16:   startmsg  = 'Durchsuche Verzeichnisse ';
[FB2]   17:   read_err  = -1;
[WD2]   18:   backslash = '\';
[Ø52]   19:   maxparm   = 1;
[ØØØ]   2Ø:
[STØ]   21: TYPE
[424]   22:   filestr   = STRING[13];
[XD3]   23:   str2      = STRING[2];
[HX2]   24:   fileinfo  = RECORD
[9N2]   25:                 name    : filestr;
[XG2]   26:                 path    : dirstr;
[ZNØ]   27:               END;
[Y35]   28:   duplicaterec = ^duplicaterec_;
[DS3]   29:   duplicaterec_ = RECORD
[A92]   3Ø:                   path : dirstr;
[WB3]   31:                   next : duplicaterec;
[ZNØ]   32:                 END;
[XR3]   33:   fdf_typ = ^fdf_typ_;
[LU2]   34:   fdf_typ_ = RECORD
[S43]   35:                descr      : fileinfo;
[ED4]   36:                duplicates : duplicaterec;
[5Ø3]   37:                next       : fdf_typ;
[ZNØ]   38:              END;
[ØØØ]   39:
[ET3]   4Ø: VAR  liste    : fdf_typ;
[3K3]   41:      hd_drive : STRING[2];
[ØØØ]   42:
[BH5]   43: FUNCTION newheaperror(VAR size:
[KP2]   44:                       WORD) : INTEGER;
```

Listing 4. Mit »fdf.pas« erkennen Sie mehrfach vorhandene Dateien ohne lange Sucharbeit

```
[MUØ]   45: BEGIN
[VU2]   46:   newheaperror:= 1;
[ZNØ]   47: END;
[ØØØ]   48:
[ØV2]   49: PROCEDURE abort;
[MUØ]   5Ø: BEGIN
[D23]   51:   WriteLn(mem_err);
[T41]   52:   Halt(1);
[ZNØ]   53: END;
[ØØØ]   54:
[KD7]   55: PROCEDURE checkfilename(VAR liste:fdf_typ;
[RH2]   56:                         path     :dirstr;
[TV2]   57:                         name     :filestr);
[NE2]   58: VAR newfdfrec,
[EE3]   59:     actualrec : fdf_typ;
[D34]   6Ø:     duplrec   : duplicaterec;
[MUØ]   61: BEGIN
[8W3]   62:   path:= path + backslash;
[GP2]   63:   New(newfdfrec);
[7Ø4]   64:   IF (newfdfrec = NIL) THEN
[F11]   65:     abort;
[2X4]   66:   newfdfrec^.descr.name:= name;
[SY4]   67:   newfdfrec^.descr.path:= path;
[BY4]   68:   newfdfrec^.duplicates:= NIL;
[ØE4]   69:   newfdfrec^.next      := NIL;
[Ø92]   7Ø:   New(duplrec);
[RM3]   71:   IF (duplrec = NIL) THEN
[F11]   72:     abort;
[RK3]   73:   duplrec^.path:= path;
[EF3]   74:   duplrec^.next:= NIL;
[A93]   75:   IF (liste = NIL) THEN
[MUØ]   76:     BEGIN
[A43]   77:       liste:= newfdfrec;
[ØWØ]   78:       Exit;
[ZNØ]   79:     END;
[T33]   8Ø:   actualrec:= liste;
[1C5]   81:   WHILE (actualrec^.next <> NIL) AND
[4G5]   82:         (actualrec^.descr.name <> name) DO
[HY4]   83:     actualrec:= actualrec^.next;
[686]   84:   IF (actualrec^.descr.name <> name) THEN
[HU4]   85:     actualrec^.next:= newfdfrec
[ZRØ]   86:   ELSE
[MUØ]   87:     BEGIN
[MØ6]   88:       WHILE (actualrec^.duplicates <> NIL)
[NBØ]   89:         DO
[S54]   9Ø:         actualrec^.duplicates:=
[L25]   91:                 actualrec^.duplicates^.next;
[5N5]   92:       actualrec^.duplicates:= duplrec;
[ZNØ]   93:     END;
[ZNØ]   94: END;
[ØØØ]   95:
[YZ4]   96: PROCEDURE ftw(path : pathstr);
[XB3]   97: VAR buf : searchrec;
[MUØ]   98: BEGIN
[UØ6]   99:   FindFirst(path + '\*.*', anyfile, buf);
[5P3]  1ØØ:   WHILE (doserror = Ø) DO
[MUØ]  1Ø1:     BEGIN
[U64]  1Ø2:       IF (buf.name[1] <> '.') THEN
[MUØ]  1Ø3:         BEGIN
[3W4]  1Ø4:           IF (buf.attr = directory) THEN
[4G3]  1Ø5:             ftw(path + backslash +
[LF1]  1Ø6:                 buf.name)
[ZRØ]  1Ø7:           ELSE
[284]  1Ø8:             checkfilename(liste,path,
[4N1]  1Ø9:                           buf.name);
[ZNØ]  11Ø:         END;
[ND2]  111:       FindNext(buf);
[ZNØ]  112:     END;
[ZNØ]  113: END;
[ØØØ]  114:
[V66]  115: PROCEDURE showduplicates(VAR liste :
[1L1]  116:                          fdf_typ);
[8D4]  117: VAR duprec : duplicaterec;
[ØP2]  118:     fsize  : LONGINT;
[AR2]  119:     fdate  : datetime;
[ØØØ]  12Ø:
[ND5]  121: FUNCTION fill(w, z : Word) : Str2;
[EX2]  122: VAR tmp : STRING;
[LØ2]  123:     p   : INTEGER;
[MUØ]  124: BEGIN
[5E2]  125:   Str(w:z, tmp);
[WM2]  126:   p:= Pos(' ', tmp);
[B42]  127:   WHILE (p > Ø) DO
[MUØ]  128:     BEGIN
[SØ2]  129:       tmp[p]:= 'Ø';
[WM2]  13Ø:       p:= Pos(' ', tmp);
[ZNØ]  131:     END;
[SU1]  132:   fill:= tmp;
[ZNØ]  133: END;
[ØØØ]  134:
[ZS6]  135: PROCEDURE fileinfos(path, name : STRING;
[SB3]  136:                     VAR size   : LONGINT;
[XK3]  137:                     VAR date   : datetime);
[3B3]  138: VAR  buf : searchrec;
[MUØ]  139: BEGIN
[MV5]  14Ø:   FindFirst(path + name, anyfile, buf);
[ZC3]  141:   IF (doserror = Ø) THEN
[MUØ]  142:     BEGIN
[HM2]  143:       size:= buf.size;
[PF4]  144:       UnpackTime(buf.time, date);
[4MØ]  145:     END
```

```
[ZRØ]  146:   ELSE
[AR2]  147:     size:= read_err;
[ZNØ]  148: END;
[ØØØ]  149:
[NA6]  15Ø: PROCEDURE writeinfos(size : LONGINT;
[4N2]  151:                      date : datetime);
[MUØ]  152: BEGIN
[ZL4]  153:   Write(' ', size, ' Bytes, vom ');
[EY1]  154:   WITH date DO
[MUØ]  155:     BEGIN
[DZ3]  156:       IF (size = read_err) THEN
[L23]  157:         WriteLn(open_err)
[ZRØ]  158:       ELSE
[HH4]  159:         WriteLn(', ',day,'.',month,'.',
[EE3]  16Ø:                 year,',  ',fill(hour,2),
[DT2]  161:                 ':',fill(min, 2),':',
[5A2]  162:                 fill(sec,2),')');
[ZNØ]  163:     END;
[ZNØ]  164: END;
[ØØØ]  165:
[MUØ]  166: BEGIN
[P31]  167:   REPEAT
[WH5]  168:     IF (liste^.duplicates <> NIL) THEN
[MUØ]  169:       BEGIN
[7U4]  17Ø:         fileinfos(liste^.descr.path,
[444]  171:                   liste^.descr.name,fsize,
[151]  172:                   fdate);
[CS5]  173:         WriteLn(liste^.descr.name, ' in:');
[LZ4]  174:         Write(''':5, liste^.descr.path);
[F74]  175:         writeinfos(fsize, fdate);
[YV4]  176:         duprec:= liste^.duplicates;
[AN3]  177:         WHILE (duprec <> NIL) DO
[MUØ]  178:           BEGIN
[614]  179:             fileinfos(duprec^.path,
[T13]  18Ø:                       liste^.descr.name,
[U52]  181:                       fsize, fdate);
[M64]  182:             Write(''':5, duprec^.path);
[F74]  183:             writeinfos(fsize, fdate);
[9T3]  184:             duprec:= duprec^.next;
[ZNØ]  185:           END;
[ZNØ]  186:       END;
[TH3]  187:     liste:= liste^.next;
[B23]  188:   UNTIL (liste = NIL);
[ZNØ]  189: END;
[ØØØ]  19Ø:
[MUØ]  191: BEGIN
[975]  192:   hd_drive := ParamStr(maxparm);
[4H4]  193:   heaperror:= @newheaperror;
[LZ1]  194:   New(liste);
[122]  195:   liste:= NIL;
[DT2]  196:   Write(startmsg);
[7K2]  197:   ftw(hd_drive);
[XF1]  198:   WriteLn;
[9V3]  199:   showduplicates(liste);
[KNØ]  2ØØ: END.
```

Listing 4. Mit »fdf.pas« erkennen Sie mehrfach vorhandene Dateien ohne lange Sucharbeit

```
[8E3]    1: { Programm: fdel.pas
[BZ6]    2:   Funktion: löscht Unterverzeichnisäste
[YY5]    3:   Sprache:  Turbo Pascal ab Version 5.Ø
[W63]    4:   Autor:    Georg Pohl
[P35]    5:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    6:
[2C2]    7: USES  Dos, Crt;
[CØ2]    8: {$I-,V-,R-}
[ØØØ]    9:
[6XØ]   1Ø: CONST
[WD2]   11:   backslash = '\';
[572]   12:   cr        = ^j^m;
[ZG2]   13:   maske     = '\*.*';
[7U2]   14:   janein    = ' (J/N) ';
[FH6]   15:   message   = 'Durchsuche Verzeichnisse'+
[7TØ]   16:               ' ... ';
[TF6]   17:   filesdel  = 'Dateien zu löschen .....: ';
[7K6]   18:   dirsdel   = 'Verzeichnisse zu löschen: ';
[L22]   19:   usage     = cr +
[GK4]   2Ø:               'Aufruf: FDEL Verzeichnis' +
[1H2]   21:               '-Name [/?]' + cr;
[4L6]   22:   unknown   = 'Unbekanntes Verzeichnis ';
[ØØØ]   23:
[STØ]   24: TYPE
[2R2]   25:   stack  = ^stack_;
[AM2]   26:   stack_ = RECORD
[HG2]   27:              name : pathstr;
[SØ2]   28:              next : stack;
[ZNØ]   29:            END;
[ØØØ]   3Ø:
[F72]   31: VAR dirstack,
[143]   32:     filestack   : stack;
[AK3]   33:     dirs,files  : LONGINT;
[ØØØ]   34:
[WU6]   35: PROCEDURE push(VAR s : stack; e : pathstr);
[CR2]   36: VAR tmp : stack;
[MUØ]   37: BEGIN
[XK1]   38:   New(tmp);
[BA2]   39:   tmp^.name:= e;
[ØD2]   4Ø:   tmp^.next:= s;
[KZ1]   41:   s        := tmp;
```

```
[ZNØ]   42: END;
[ØØØ]   43:
[HP4]   44: PROCEDURE pop(VAR s : stack;
[DU2]   45:                   VAR e : pathstr);
[6D3]   46: VAR scratch : stack;
[ØØØ]   47:
[MUØ]   48: BEGIN
[412]   49:   scratch:= s;
[5Ø3]   5Ø:   e:= scratch^.name;
[U23]   51:   s:= scratch^.next;
[UZ2]   52:   Dispose(scratch);
[ZNØ]   53: END;
[ØØØ]   54:
[DK3]   55: PROCEDURE initstack;
[MUØ]   56: BEGIN
[HN2]   57:   New(filestack);
[KD2]   58:   New(dirstack);
[ZR2]   59:   filestack:= NIL;
[4N2]   6Ø:   dirstack := NIL;
[NW1]   61:   files    := Ø;
[HU1]   62:   dirs     := Ø;
[ZNØ]   63: END;
[ØØØ]   64:
[HFC]   65: VAR   startdir,homedir   : pathstr;
                                              { gerufe
        n wurde }
[983]   66:       x, y               : INTEGER;
[1W3]   67:       buf                : searchrec;
[DV3]   68:       mitfrage           : BOOLEAN;
[ØØØ]   69:
[SG4]   7Ø: FUNCTION yes_no : BOOLEAN;
[WA2]   71: VAR ch   : CHAR;
[ØS2]   72:     regs : REGISTERS;
[MUØ]   73: BEGIN
[F85]   74:   { Tastaturbuffer entleeren ... }
[6A2]   75:   regs.ax:= $ØCØØ;
[N14]   76:   MsDos(REGISTERS(regs));
[ZG3]   77:   ch:= UpCase(ReadKey);
[2R2]   78:   yes_no:= ch = 'J';
[ZNØ]   79: END;
[ØØØ]   8Ø:
[YZ4]   81: PROCEDURE ftw(path : pathstr);
[XB3]   82: VAR buf : searchrec;
[MUØ]   83: BEGIN
[7L6]   84:   IF (path[Length(path)] = backslash) THEN
[UP4]   85:     Delete(path, Length(path), 1);
[L66]   86:   FindFirst(path + maske, anyfile, buf);
[5P3]   87:   WHILE (doserror = Ø) DO
[MUØ]   88:     BEGIN
[U64]   89:       IF (buf.name[1] <> '.') THEN
[MUØ]   9Ø:         BEGIN
[3W4]   91:           IF (buf.attr = directory) THEN
[MUØ]   92:             BEGIN
[MB3]   93:               push(dirstack,path +
[ZE3]   94:               backslash + buf.name);
[HN1]   95:               Inc(dirs);
[4G3]   96:               ftw(path + backslash +
[LF1]   97:                   buf.name)
[4MØ]   98:             END
[ZRØ]   99:           ELSE
[MUØ]  1ØØ:             BEGIN
[MK3]  1Ø1:               push(filestack,path +
[ZE3]  1Ø2:                    backslash + buf.name);
[TW1]  1Ø3:               Inc(files);
[ZNØ]  1Ø4:             END;
[ZNØ]  1Ø5:         END;
[ND2]  1Ø6:       FindNext(buf);
[ZNØ]  1Ø7:     END;
[ZNØ]  1Ø8: END;
[ØØØ]  1Ø9:
[TC3]  11Ø: PROCEDURE delfiles;
[873]  111: VAR name : pathstr;
[NM1]  112:     fp   : FILE;
[4Y1]  113:     attr : WORD;
[FD2]  114:     del  : BOOLEAN;
[MUØ]  115: BEGIN
[9C4]  116:   WHILE (filestack <> NIL) DO
[MUØ]  117:     BEGIN
[HD3]  118:       GotoXY(x, y); ClrEoL;
[NC3]  119:       pop(filestack,name);
[ZY1]  12Ø:       Write(name);
[XW2]  121:       IF (mitfrage) THEN
[MUØ]  122:         BEGIN
[7B2]  123:           Write(janein);
[482]  124:           del:= yes_no;
[ZNØ]  125:         END;
[W83]  126:       IF (mitfrage AND del)
[6P3]  127:         OR (NOT mitfrage) THEN
[MUØ]  128:         BEGIN
[WL2]  129:           Assign(fp, name);
[LA3]  13Ø:           GetFAttr(fp, attr);
[BX1]  131:           IF (attr AND
[YX4]  132:             (readonly OR hidden) <> Ø) THEN
[MUØ]  133:             BEGIN
[HT4]  134:               attr:=attr AND (NOT(readonly)
[LS2]  135:                     AND NOT(hidden));
[W63]  136:               SetFAttr(fp,attr);
[ZNØ]  137:             END;
[FM1]  138:           Reset(fp);
[4K1]  139:           Erase(fp);
```

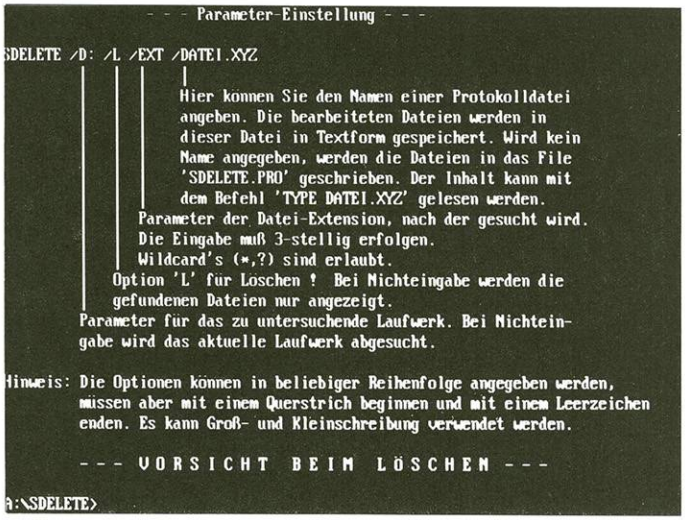
```
[6D3]  14Ø:           IF (IOResult = Ø) THEN
[6L1]  141:             Close(fp);
[ZNØ]  142:         END;
[ZNØ]  143:     END;
[ZNØ]  144: END;
[ØØØ]  145:
[143]  146: PROCEDURE deldirs;
[873]  147: VAR name : pathstr;
[FD2]  148:     del  : BOOLEAN;
[MUØ]  149: BEGIN
[K14]  15Ø:   WHILE (dirstack <> NIL) DO
[MUØ]  151:     BEGIN
[842]  152:       GotoXY(x, y);
[5B1]  153:       ClrEoL;
[E93]  154:       pop(dirstack, name);
[ZY1]  155:       Write(name);
[XW2]  156:       IF (mitfrage) THEN
[MUØ]  157:         BEGIN
[7B2]  158:           Write(janein);
[482]  159:           del:= yes_no;
[ZNØ]  16Ø:         END;
[W83]  161:       IF (mitfrage AND del)
[6P3]  162:         OR (NOT mitfrage) THEN
[MUØ]  163:         BEGIN
[1Y1]  164:           RmDir(name);
[LM3]  165:           IF (IOResult <> Ø) THEN
[MUØ]  166:             BEGIN
[F34]  167:               WriteLn(' --> not empty');
[1Z1]  168:               y:= WhereY;
[ZNØ]  169:             END;
[ZNØ]  17Ø:         END;
[ZNØ]  171:     END;
[ZNØ]  172: END;
[ØØØ]  173:
[YT3]  174: PROCEDURE checkparams;
[283]  175: VAR xdir : pathstr;
[792]  176:     ok   : BOOLEAN;
[MUØ]  177: BEGIN
[HT2]  178:   mitfrage:= FALSE;
[CA3]  179:   ok:= ParamCount >= 1;
[HT1]  18Ø:   IF (ok) THEN
[MUØ]  181:     BEGIN
[1W3]  182:       startdir:= ParamStr(1);
[XN3]  183:       IF (ParamCount = 2) THEN
[MUØ]  184:         BEGIN
[GE3]  185:           ok:= ParamStr(2) = '/?';
[M82]  186:           mitfrage:= ok;
[ZNØ]  187:         END;
[ZNØ]  188:     END;
[AH2]  189:   IF (NOT ok) THEN
[MUØ]  19Ø:     BEGIN
[9P2]  191:       WriteLn(usage);
[T41]  192:       Halt(1);
[ZNØ]  193:     END;
[X13]  194:   GetDir(Ø, homedir);
[6R2]  195:   ChDir(startdir);
[LM3]  196:   IF (IOResult <> Ø) THEN
[MUØ]  197:     BEGIN
[DY4]  198:       WriteLn(unknown, startdir);
[FG2]  199:       ChDir(homedir);
[T41]  2ØØ:       Halt(1);
[ZNØ]  2Ø1:     END;
[M83]  2Ø2:   GetDir(Ø, startdir);
[WR6]  2Ø3:   xdir:= Copy(homedir,1,Length(startdir));
[EY3]  2Ø4:   IF (startdir = xdir) THEN
[3M4]  2Ø5:     IF (homedir > startdir) THEN
[746]  2Ø6:       IF (homedir[Succ(Length(startdir))]
[3T2]  2Ø7:         = backslash) THEN
[NE3]  2Ø8:         homedir:= startdir;
[FG2]  2Ø9:   ChDir(homedir);
[ZNØ]  21Ø: END;
[ØØØ]  211:
[MUØ]  212: BEGIN
[4T1]  213:   initstack;
[142]  214:   checkparams;
[843]  215:   Write(cr, message);
[UB6]  216:   { Dateien u. Verzeichnisse einlesen }
[HF2]  217:   ftw(startdir);
[XF1]  218:   WriteLn;
[384]  219:   WriteLn(filesdel, files);
[A14]  22Ø:   WriteLn(dirsdel, dirs);
[D24]  221:   IF (filestack <> NIL) THEN
[MUØ]  222:     BEGIN
[PN2]  223:       Write(cr, 'del   ');
[WY1]  224:       x:= WhereX;
[1Z1]  225:       y:= WhereY;
[ØH1]  226:       delfiles;
[ZNØ]  227:     END;
[5Z3]  228:   IF (dirstack <> NIL) THEN
[MUØ]  229:     BEGIN
[KW2]  23Ø:       Write(cr, 'rmdir ');
[WY1]  231:       x:= WhereX;
[1Z1]  232:       y:= WhereY;
[UE1]  233:       deldirs;
[ZNØ]  234:     END;
[KNØ]  235: END.
```

Listing 5. Dank »fdel.pas« ist das mühsame Verzeichnislöschen beendet. Ab jetzt reicht ein Befehl aus

Turbo Basic

Festplatte jetzt ohne Ballast

Dateien mit Erweiterungen wie beispielsweise ».bak« oder ».sik« kosten unnötigen Speicherplatz. Von Zeit zu Zeit sollten diese gelöscht werden. Es ist jedoch ein mühseliges Unterfangen, wenn man die Verzeichnisse nacheinander nach »Müll« absuchen muß. Deshalb sollten wir diese Arbeit dem Computer überlassen.



Durch ausführliche Hinweise beim Aufruf fällt die Bedienung sehr leicht

Mit »sdelete.bas« (Listing) läuft diese Prozedur programmgesteuert ab. Das Programm durchsucht die gesamte Festplatte nach vorgewählten Erweiterungen und schreibt darüber Protokoll. Gelöscht wird erst dann, wenn man sich endgültig dazu entschließt.

Das Programm ist besonders für Festplatten mit vielen Verzeichnissen und Unterverzeichnissen sinnvoll. Man wundert sich oft, wieviel Speicherplatz auf der Festplatte im Laufe der Zeit durch nicht mehr benötigte Dateien verloren geht. In diesem Fall sorgt »sdelete« für einen vollständigen Überblick und beseitigt auf Wunsch allen unnötigen Ballast.

Das Programm läßt sich in mehreren Stufen aufrufen:

Beim Aufruf von »sdelete« ohne Parameter wird die Anwendung erklärt (Bild).

```
sdelete /d: /ext
```

sucht Laufwerk D nach der Erweiterung »ext« ab und schreibt eine Liste der gefundenen Dateien in die Default-Datei »sdelete.pro«. Wird kein Laufwerk angegeben, nimmt das Programm das aktuelle als das zu durchsuchende an.

```
sdelete /d: /ext /muell.xyz
```

arbeitet wie das vorher beschriebene Beispiel. Die Liste wird jedoch in der angegebenen Datei »muell.xyz« abgelegt.

```
sdelete /d: /l /ext /muell.xyz
```

löscht die gefundenen Dateien auf dem gesamten Laufwerk.

Dabei wird wieder ein Protokoll in der Datei »muell.xyz« angelegt (oder beim Weglassen in der Default-Datei »sdelete.pro«).

Die Optionen können in beliebiger Reihenfolge eingegeben werden. Sie müssen jedoch alle mit einem »/« beginnen und mit einem Leerschritt enden. Das Programm wertet sowohl Groß- als auch Kleinschreibung aus.

Das angelegte Protokoll über den Löschvorgang dient unter anderem dazu, versehentliches Löschen wieder rückgängig zu machen. Bevor man nach dem Abschluß der »Löscharbeiten« mit dem Rechner weiterarbeitet, sollte das Protokoll durchgelesen werden. Irrtümlich gelöschte Dateien können dann mit bekannten Tools wie Norton Utilities oder PC-Tools wieder regeneriert werden.

»sdelete.bas« kann mit Turbo Basic 1.0 oder höher kompiliert werden. Um »sdelete.exe« aus jedem Verzeichnis aufrufen zu können, sollte eine entsprechende Pfadanweisung in der »autoexec.bat« stehen.

Das bedeutet, Sie müssen »sdelete.exe« in ein Verzeichnis kopieren, auf das mittels der Pfad-Anweisung verwiesen wird, beispielsweise »\dos« oder »\tools«.

Sie können sich die Arbeit auch dadurch vereinfachen, indem Sie für den »Hausputz« eine Bat-Datei generieren. Diese kann dann mit einem einzigen Befehl aufgerufen werden.

(Freddy Leberkern/hs)

```
[K83]    1: 'Programm: xdel.bas
[2Ø5]    2: 'Funktion: globales Lschen
[ØF3]    3: 'Sprache:  Turbo Basic
[7D4]    4: 'Autor:    Freddy Leberkern
[EP4]    5: '(C)1991 DMV Widuch GmbH & Co. KG
[ØØØ]    6:
[PNØ]    7: CLS
[FA3]    8: ON ERROR GOTO fehler
[ØØØ]    9:
[X15]   1Ø: '--------- Variablen  definieren ---
[ØØØ]   11:
[XZ3]   12: ax      = 1 : bx      = 2  : cx = 3
[LØ4]   13: dx      = 4 : di      = 6  : ds = 8
[E71]   14: kb&     = Ø
[E45]   15: flag%   = Ø  : index% = Ø  : anzdir% = Ø
[825]   16: drive%  = Ø  : opt%   = Ø  : clflag% = Ø
[1N4]   17: tmp$    = "" : dir$   = "" : ext$     = ""
[HØ4]   18: drive$  = ""  : prodatei$ = ""
[MY5]   19: text1$  = "Folgende Datei(en) wurde(n) "
[M53]   2Ø: text2$  = "angezeigt."
[BC3]   21: text3$  = "gelscht."
[ØØØ]   22:
[RP2]   23: DIM direc$ (Ø:1ØØØ)
[D42]   24: DIM fcb% (Ø:42)
[NP3]   25: fcbseg& = VARSEG (fcb%(Ø))
[NT3]   26: fcbofs& = VARPTR (fcb%(Ø))
```

```
[ØØØ]   27:
[Z52]   28: DIM dta% (Ø:63)
[7S3]   29: dtaseg& = VARSEG (dta%(Ø))
[7V3]   3Ø: dtaofs& = VARPTR (dta%(Ø))
[ØØØ]   31:
[SZ5]   32: '--- aktuelles Laufwerk ermitteln ----
[ØØØ]   33:
[UØ2]   34: REG ax, &H19ØØ
[N53]   35: CALL INTERRUPT &H21
[K33]   36: defdrive% = REG(ax)
[AM5]   37: defdrive$ = CHR$(defdrive% - 6335) + ":"
[ØØØ]   38:
[ØA6]   39: '--- aktuelles Verzeichnis ermitteln --
[ØØØ]   4Ø:
[P42]   41: GOSUB setdta
[412]   42: REG ax, &H47ØØ
[LØ2]   43: REG dx, &HØØØØ
[KE2]   44: REG ds, dtaseg&
[PE2]   45: REG di, dtaofs&
[N53]   46: CALL INTERRUPT &H21
[ØN2]   47: DEF SEG = dtaseg&
[V31]   48: i%      = Ø
[PP2]   49: WHILE PEEK(i%) <> Ø
```

Listing. Mit »sdelete.bas« wird das Löschen von Dateigruppen kinderleicht

```
[B94]   5Ø:   pfad$ = pfad$ + CHR$(PEEK(i%))
[941]   51:   INCR i%
[TRØ]   52: WEND
[YL4]   53: aktdir$ = defdrive$ + "\" + pfad$
[ØØØ]   54:
[FL6]   55: '--- Optionen einlesen und berprfen ---
[ØØØ]   56:
[PK4]   57: parameter$ = ucase$(command$)
[WF4]   58: parameter$ = parameter$ + " /  "
[ØØØ]   59:
[3B4]   6Ø: FOR i% = 1 TO LEN(parameter$)
[XX4]   61:   IF MID$(parameter$,i%,1) = "/" THEN
[5F1]   62:     INCR opt%
[YG2]   63:     fcb%(opt%) = i% + 1
[8Ø1]   64:   END IF
[E61]   65: NEXT i%
[ØE1]   66: DECR opt%
[ØØØ]   67:
[5B6]   68: IF opt% < 1 OR opt% > 4 THEN GOTO parerror
[7F2]   69: FOR i% = 1 TO opt%
[DL5]   7Ø:   opt$(i%) = MID$(parameter$, fcb%(i%),_
[983]   71:                fcb%(i% + 1) - fcb%(i%) - 2)
[E61]   72: NEXT i%
[ØØØ]   73:
[7F2]   74: FOR i% = 1 TO opt%
[UC3]   75:   IF LEN(opt$(i%)) = 1 THEN
[AX2]   76:     IF opt$(i%) = "L" THEN
[RL1]   77:       clflag% = 1
[ZRØ]   78:     ELSE
[VA2]   79:       GOTO parerror
[8Ø1]   8Ø:     END IF
[8Ø1]   81:   END IF
[VC3]   82:   IF LEN(opt$(i%)) = 2 THEN
[2A4]   83:     IF MID$(opt$(i%),2,1) = ":" THEN
[DN2]   84:       dir$      = opt$(i%)
[H82]   85:       drive$    = dir$
[8Y2]   86:       direc$(Ø) = dir$ + "\"
[4N4]   87:       drive%    = ASC(MID$(dir$,1,1))-65
[ZRØ]   88:     ELSE
[VA2]   89:      GOTO parerror
[8Ø1]   9Ø:     END IF
[8Ø1]   91:   END IF
[WP3]   92:   IF     LEN(opt$(i%)) = 3 THEN
[562]   93:     ext$ = opt$(i%)
[Y44]   94:   ELSEIF LEN(opt$(i%)) > 3 THEN
[R43]   95:     prodatei$ = opt$(i%)
[8Ø1]   96:   END IF
[E61]   97: NEXT i%
[ØØØ]   98:
[TD5]   99: IF LEN(ext$) = Ø THEN GOTO parerror
[7Y3]  1ØØ: IF LEN(prodatei$) = Ø THEN
[234]  1Ø1:   prodatei$ = "SDELETE.PRO"
[8Ø1]  1Ø2: END IF
[ØØØ]  1Ø3:
[NS2]  1Ø4: IF drive$ = "" THEN
[T23]  1Ø5:   drive$    = defdrive$
[UY2]  1Ø6:   dir$      = defdrive$
[8Y2]  1Ø7:   direc$(Ø) = dir$ + "\"
[Y23]  1Ø8:   drive%    = defdrive%
[8Ø1]  1Ø9: END IF
[ØØØ]  11Ø:
[AB3]  111: OPEN "O", #1, prodatei$
[Ø51]  112: CLOSE #1
[KC2]  113: KILL prodatei$
[AB3]  114: OPEN "O", #1, prodatei$
[591]  115: PRINT #1,
[HA6]  116: IF clflag% = Ø THEN PRINT #1, text1$+text2$
[RA6]  117: IF clflag% = 1 THEN PRINT #1, text1$+text3$
[591]  118: PRINT #1,
[ØØØ]  119:
[1F3]  12Ø: CALL setdrive(drive%)
[V92]  121: CHDIR direc$(Ø)
[K23]  122: CALL fredskspc(kb&)
[NZ1]  123: vorher& = kb&
[ØØØ]  124:
[U27]  125: '-- Hauptdirectorys suchen und speichern --
[ØØØ]  126:
[S73]  127: CALL holedir(1,flag%)
[9V4]  128: IF flag% = Ø THEN GOTO loeschen
[H92]  129: WHILE flag% <> Ø
[3M4]  13Ø:   IF tmp$ <> "." AND tmp$ <> ".." THEN
[Z12]  131:     incr anzdir%
[T65]  132:     direc$(anzdir%) = drive$ + "\" + tmp$
[8Ø1]  133:   END IF
[V73]  134:   CALL holedir(2,flag%)
[TRØ]  135: WEND
[ØØØ]  136:
[8Z6]  137: '-- Zieldateien in Nebenverzeichn. löschen
[ØØØ]  138:
[MT2]  139: IF anzdir% > Ø THEN
[E21]  14Ø: start:
[7Y2]  141:   counter% = anzdir%
[FM4]  142:   FOR i% = 1 + index% TO counter%
[ZC2]  143:     CHDIR direc$(i%)
[S73]  144:     CALL holedir(1,flag%)
[H92]  145:     WHILE flag% <> Ø
[V73]  146:       CALL holedir(2,flag%)
[WL4]  147:       IF tmp$ <> "." AND tmp$ <> ".." AND_
[X52]  148:          flag% = 1   THEN
[Z12]  149:         INCR anzdir%
```

```
[G25]  15Ø:         direc$(anzdir%)=direc$(i%)+"\"+tmp$
[8Ø1]  151:       END IF
[TRØ]  152:     WEND
[E61]  153:   NEXT i%
[724]  154:   IF anzdir% > counter% THEN
[FM2]  155:     index% = counter%
[4W1]  156:     GOTO start
[8Ø1]  157:   END IF
[ØØØ]  158:
[4F6]  159: '-- Verzeichn. alphabetisch sortieren --
[ØØØ]  16Ø:
[BC3]  161:   FOR z% = Ø TO anzdir% - 1
[9B3]  162:     FOR i% = Ø TO anzdir% - 1
[UM4]  163:       IF direc$(i%) > direc$(i%+1) THEN
[UØ4]  164:         SWAP direc$(i%), direc$(i%+1)
[8Ø1]  165:       END IF
[E61]  166:     NEXT i%
[181]  167:   NEXT z%
[8Ø1]  168: END IF
[ØØØ]  169:
[MK6]  17Ø: '-- nach zu löschenden Dateien suchen ---
[ØØØ]  171:
[ØH1]  172: loeschen:
[LY2]  173: FOR i% = Ø TO anzdir%
[ZC2]  174:   CHDIR direc$(i%)
[614]  175:   CALL holedat(1,flag%,ext$)
[H92]  176:     WHILE flag% <> Ø
[PF2]  177:       IF tmp$ <> "" THEN
[RX1]  178:         IF i% = Ø THEN
[PK4]  179:           datei$=MID$(direc$(i%),1,2)+tmp$
[ZRØ]  18Ø:         ELSE
[A74]  181:           datei$ = direc$(i%) + "\" + tmp$
[8Ø1]  182:         END IF
[UR2]  183:         IF clflag% = 1 THEN
[XW1]  184:           KILL  datei$
[Ø12]  185:           PRINT datei$
[NC2]  186:           PRINT #1, datei$
[8Ø1]  187:         END IF
[914]  188:         CALL holedat(2,flag%,ext$)
[8Ø1]  189:       END IF
[TRØ]  19Ø:     WEND
[E61]  191: NEXT i%
[ØØØ]  192:
[XF6]  193: '-- Meldung freier Diskbereich ausgeben
[ØØØ]  194:
[XTØ]  195: ende:
[K23]  196: CALL fredskspc(kb&)
[S12]  197: nachher& = kb&
[GXØ]  198: PRINT
[146]  199: PRINT "Freier Bereich auf Disk  "drive$;
[5C6]  2ØØ: PRINT USING" vorher: ####### KB's";vorher&
[146]  2Ø1: PRINT "Freier Bereich auf Disk  "drive$;
[SB6]  2Ø2: PRINT USING" jetzt : ####### KB's";nachher&
[ØØØ]  2Ø3:
[GF4]  2Ø4: IF nachher& - vorher& > Ø THEN
[NX8]  2Ø5:   PRINT TAB(38)"═══════════"
[Z64]  2Ø6:   summe& = nachher& - vorher&
[EW4]  2Ø7:   p1$ = "       Gewonnener Disk – "
[B82]  2Ø8:   p2$ = "Bereich   "
[AX1]  2Ø9:   pg$ = p1$ + p2$
[4M5]  21Ø:   PRINT USING pg$+": ####### KB's";summe&
[GXØ]  211:   PRINT
[8Ø1]  212: END IF
[ØØØ]  213:
[V92]  214: CHDIR direc$(Ø)
[PY3]  215: CALL setdrive(defdrive%)
[W12]  216: CHDIR aktdir$
[Ø51]  217: CLOSE #1
[ØØØ]  218:
[4MØ]  219: END
[ØØØ]  22Ø:
[PK4]  221: ' ---- U n t e r p r o g r a m m e ----
[ØØØ]  222:
[ML6]  223: ' ---- Platten - Transferbereich festlegen
[ØØØ]  224:
[WA1]  225: setdta:
[212]  226:   REG ax, &H1AØØ
[KE2]  227:   REG ds, dtaseg&
[3G2]  228:   REG dx, dtaofs&
[N53]  229:   CALL INTERRUPT &H21
[651]  23Ø: RETURN
[ØØØ]  231:
[Y54]  232: ' ---- Laufwerk wechseln ----
[ØØØ]  233:
[TZ2]  234: SUB setdrive(lw%)
[LW1]  235:   SHARED ax, dx
[212]  236:   REG ax, &HØEØØ
[ØU1]  237:   REG dx, lw%
[N53]  238:   CALL INTERRUPT &H21
[HA1]  239: END SUB
[ØØØ]  24Ø:
[ER5]  241: ' --- nach Verzeichnissen suchen -----
[ØØØ]  242:
[ZV3]  243: SUB holedir (art%,flag%)
[ØM5]  244:   SHARED ax, cx, dx, ds, dtaseg&, dtaofs&
[TP2]  245:   SHARED dta%(), tmp$
[P42]  246:   GOSUB setdta
[ØØØ]  247:
```

Listing. Mit »sdelete.bas« wird das Löschen von Dateigruppen kinderleicht

```
[T82]  248:    IF art% = 1 THEN
[ØN2]  249:      DEF SEG = dtaseg&
[562]  25Ø:      FOR i% =  Ø TO 1Ø
[ZU1]  251:        POKE i%, &H2A
[E61]  252:      NEXT i%
[582]  253:      FOR i% = 11 TO 43
[VT1]  254:        POKE i%, &HØØ
[E61]  255:      NEXT i%
[K12]  256:      REG ax, &H4EØØ
[LØ2]  257:      REG cx, &HØØ1Ø
[KE2]  258:      REG ds, dtaseg&
[3G2]  259:      REG dx, dtaofs&
[N53]  26Ø:      CALL interrupt &H21
[8Ø1]  261:    END IF
[ØØØ]  262:
[X82]  263:    IF art% = 2 THEN
[ØN2]  264:      DEF SEG = dtaseg&
[L12]  265:      REG ax, &H4FØØ
[N53]  266:      CALL INTERRUPT &H21
[8Ø1]  267:    END IF
[ØØØ]  268:
[655]  269:    IF REG(ax) = Ø AND PEEK(21) = &H1Ø THEN
[V61]  27Ø:      flag% = 1
[HA1]  271:      tmp$  = ""
[E82]  272:      FOR i% = 3Ø TO 37
[794]  273:        IF PEEK(i%) = Ø THEN EXIT SUB
[XV3]  274:        tmp$ = tmp$ + CHR$(PEEK(i%))
[E61]  275:      NEXT i%
[ZRØ]  276:    ELSE
[T61]  277:      flag% = Ø
[HA1]  278:      tmp$  = ""
[TE1]  279:      EXIT SUB
[8Ø1]  28Ø:    END IF
[ØØØ]  281:
[HA1]  282: END SUB
[ØØØ]  283:
[PN6]  284: ' --- suchen nach eingegebener Erweiterung
[ØØØ]  285:
[AP4]  286: SUB holedat (art%,flag%,ext$)
[146]  287:   SHARED ax,cx,dx,ds,fcbseg&,fcbofs&,tmp$
[FP5]  288:   SHARED dtaseg&, dtaofs&, dta%(), fcb%()
[P42]  289:   GOSUB setdta
[T82]  29Ø:   IF art% = 1 THEN
[TM2]  291:     DEF SEG = fcbseg&
[KN1]  292:     POKE Ø, &HØØ
[412]  293:     FOR i% = 1 TO 8
[ZU1]  294:       POKE i%, &H2A
[E61]  295:     NEXT i%
[SX3]  296:     POKE  9, ASC(MID$(ext$,1,1))
[8Y3]  297:     POKE 1Ø, ASC(MID$(ext$,2,1))
[DY3]  298:     POKE 11, ASC(MID$(ext$,3,1))
[K82]  299:     FOR i% = 12 TO 38
[VT1]  3ØØ:       POKE i%, &HØØ
[E61]  3Ø1:     NEXT i%
[KØ2]  3Ø2:     REG ax, &H11ØØ
[7D2]  3Ø3:     REG ds, fcbseg&
[SE2]  3Ø4:     REG dx, fcbofs&
[N53]  3Ø5:     CALL interrupt &H21
[8Ø1]  3Ø6:   END IF
[ØØØ]  3Ø7:
[X82]  3Ø8:   IF art% = 2 THEN
[7D2]  3Ø9:     REG ds, fcbseg&
[SE2]  31Ø:     REG dx, fcbofs&
[LØ2]  311:     REG ax, &H12ØØ
[N53]  312:     CALL INTERRUPT &H21
[8Ø1]  313:   END IF
[ØØØ]  314:
[YY3]  315:   fehler% = REG(ax) AND &HØØFF
[G61]  316:   tmp$ = ""
[CS2]  317:   IF fehler% = Ø THEN
[ØN2]  318:     DEF SEG = dtaseg&
[3D1]  319:     flag%   = 1
[412]  32Ø:     FOR i% = 1 TO 8
[CC4]  321:       IF PEEK(i%) = 32 THEN GOTO ext
[XV3]  322:       tmp$ = tmp$ + CHR$(PEEK(i%))
[E61]  323:     NEXT i%
[WSØ]  324: ext:
[852]  325:     tmp$ = tmp$ + "."
[H22]  326:     FOR i% = 9 TO 11
[VX2]  327:       IF PEEK(i%) = 32 THEN
[TV3]  328:         IF RIGHT$(tmp$,1) = "." THEN
[ZN4]  329:           tmp$ = LEFT$(tmp$,LEN(tmp$) - 1)
[TE1]  33Ø:           EXIT SUB
[8Ø1]  331:         END IF
[8Ø1]  332:       END IF
[XV3]  333:       tmp$ = tmp$ + CHR$(PEEK(i%))
[E61]  334:     NEXT i%
[ZRØ]  335:   ELSE
[T61]  336:     FLAG% = Ø
[TE1]  337:     EXIT SUB
[8Ø1]  338:   END IF
[HA1]  339: END SUB
[ØØØ]  34Ø:
[A76]  341: ' --- Freier Platz auf der Festplatte ----
[ØØØ]  342:
[XZ2]  343: SUB fredskspc(kb&)
[E14]  344:   SHARED ax, bx, cx, dx, drive%
[ZØ2]  345:   REG ax, &H36ØØ
[YL2]  346:   REG dx, drive% + 1
[N53]  347:   CALL INTERRUPT &H21
[FT1]  348:   kb& = REG(ax)
```

```
[RN2]  349:   kb& = kb& * REG(bx)
[BC3]  35Ø:   kb& = kb& * REG(cx) / 1Ø24
[HA1]  351: END SUB
[ØØØ]  352:
[7R6]  353: ' -- Fehlerbehandlung falsche Parameter --
[ØØØ]  354:
[CM1]  355: parerror:
[GXØ]  356: PRINT
[W42]  357: PRINT TAB(18);
[EZ5]  358: PRINT "- - - Parameter-Einstellung - - -"
[GXØ]  359: PRINT
[5R5]  36Ø: PRINT "SDELETE /D: /L /EXT /DATEI.XYZ"
[3X3]  361: PRINT "          |   |   |   |"
[PX5]  362: PRINT "          |   |   |   Hier können";
[P77]  363: PRINT " Sie den Namen einer Protokolldatei"
[7P5]  364: PRINT "          |   |   |   angeben. Die";
[RD6]  365: PRINT " bearbeiteten Dateien werden in "
[NN5]  366: PRINT "          |   |   |   dieser Datei";
[Ø77]  367: PRINT " in Textform gespeichert. Wird kein"
[WH5]  368: PRINT "          |   |   |   Name angege";
[ZH6]  369: PRINT "ben, werden die Dateien in das File"
[KZ5]  37Ø: PRINT "          |   |   |   'SDELETE.PRO'";
[HR6]  371: PRINT " geschrieben. Der Inhalt kann mit"
[9B5]  372: PRINT "          |   |   |   dem Befehl";
[AS6]  373: PRINT " 'TYPE DATEI.XYZ' gelesen werden."
[GA5]  374: PRINT "          |   |   Parameter der";
[GØ7]  375: PRINT " Datei-Extension, nach der gesucht";
[F42]  376: PRINT " wird."
[RU5]  377: PRINT "          |   |   Die Eingabe muß";
[YD4]  378: PRINT " 3-stellig erfolgen."
[ME5]  379: PRINT "          |   |   Wildcard's (*,?)";
[HC3]  38Ø: PRINT " sind erlaubt."
[MX4]  381: PRINT "          |   Option 'L' für";
[ZW5]  382: PRINT " Löschen !  Bei Nichteingabe";
[2Z2]  383: PRINT " werden die"
[WL5]  384: PRINT "          |   gefundenen Dateien";
[GT3]  385: PRINT " nur angezeigt."
[5F5]  386: PRINT "          Parameter für das zu";
[M66]  387: PRINT " untersuchende Laufwerk. Bei";
[CT2]  388: PRINT " Nichtein-"
[GK5]  389: PRINT "          gabe wird das aktuelle";
[FM4]  39Ø: PRINT " Laufwerk abgesucht."
[GXØ]  391: PRINT
[6K6]  392: PRINT "Hinweis: Die Optionen können";
[PN5]  393: PRINT " in beliebiger Reihenfolge";
[W24]  394: PRINT " angegeben werden,"
[2Z5]  395: PRINT "          müssen aber mit einem";
[X87]  396: PRINT " Querstrich beginnen und mit einem";
[893]  397: PRINT " Leerzeichen"
[YC6]  398: PRINT "          enden. Es kann Groß- und";
[127]  399: PRINT " Kleinschreibung verwendet werden."
[GXØ]  4ØØ: PRINT
[7H4]  4Ø1: PRINT "          - - -  V O R S I C H T";
[VB5]  4Ø2: PRINT "   B E I M   L Ö S C H E N  - - -"
[4MØ]  4Ø3: END
[ØØØ]  4Ø4:
[YU5]  4Ø5: '----  allgemeine Fehlerbehandlung ----
[F91]  4Ø6: fehler:
[N33]  4Ø7: h1$ = "" : h2$ = "" : h3$ = ""
[KM1]  4Ø8: fzeile$ = ""
[6N2]  4Ø9: SELECT CASE ERR
[K71]  41Ø:   CASE = 75
[B22]  411:     PRINT      "  *  ";
[Ø42]  412:     PRINT #1, "  *  ";
[D12]  413:     RESUME next
[G71]  414:   CASE = 55
[LT4]  415:     h1$ = " Datei ist offen - offene"
[W55]  416:     h2$ = " Datei kann nicht gelöscht"
[Ø62]  417:     h3$ = " werden! "
[971]  418:   CASE = 61
[VB5]  419:     h1$ = " Der Datenträger ist voll"
[PL5]  42Ø:     h2$ = " !!! --> überflüssige Dateien"
[AL3]  421:     h3$ = " löschen              "
[871]  422:   CASE = 7Ø
[VØ5]  423:     h1$ = "  Die Diskette ist schreib"
[GZ3]  424:     h2$ = "geschützt !!!            "
[1K2]  425:     h3$ = "                         "
[C71]  426:   CASE = 72
[HP4]  427:     h1$ = "    Spur/Sektor-Fehler am"
[925]  428:     h2$ = " angewählten Laufwerk !!!"
[S63]  429:     h3$ = "  (CHKDSK)           "
[M71]  43Ø:   CASE = 76
[1B5]  431:     h1$ = " Drive not ready !  Diskette"
[7U4]  432:     h2$ = " einlegen / Verriegelung"
[473]  433:     h3$ = " schließen "
[GS1]  434: END SELECT
[ØØØ]  435:
[PNØ]  436: CLS
[3WØ]  437: CLOSE
[M63]  438: fzeile$ = h1$ + h2$ + h3$
[ØS2]  439:   IF fzeile$ = "" THEN
[LZ4]  44Ø:     PRINT "Turbo Basic Fehler "err" !"
[ZRØ]  441:   ELSE
[452]  442:     PRINT fzeile$
[8Ø1]  443:   END IF
[PY3]  444: CALL setdrive(defdrive%)
[W12]  445: CHDIR aktdir$
[4MØ]  446: END
```

Listing. Mit »sdelete.bas« wird das Löschen von Dateigruppen kinderleicht

Ein starkes Heft

Im ABO



Es lohnt sich!

DOS und DOS Extra
im Kombi-Abo mit Preisvorteil.

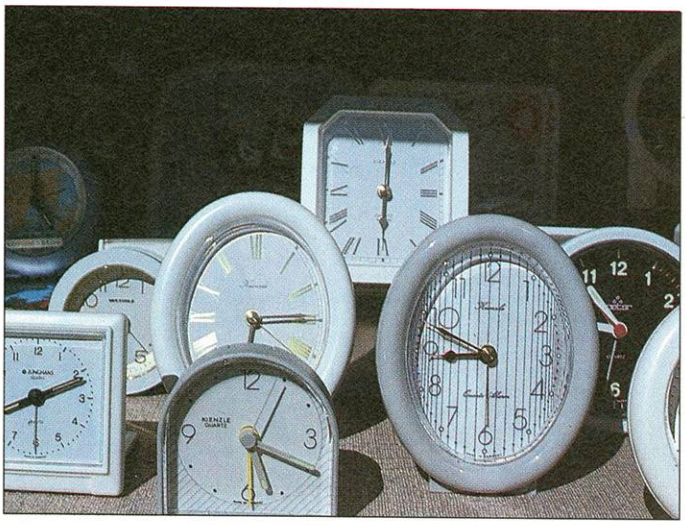
Bitte benutzen Sie die Bestellkarte.

DMV-Verlag • Postfach 250 • 3440 Eschwege

DMV Daten- und Medienverlag

MASM ab Version 4.0

Angezapfte Hardware-Uhr



Wenn Sie sofort sagen können, daß der 1. 1. 1980 ein Dienstag war, wenn all Ihre eigenen Programme und Texte laut Dateieintrag auf dem PC angeblich an diesem Tag – so zwischen Mitternacht und morgens – entstanden sind, dann haben Sie wahrscheinlich ein Problem mit der Hardware-Uhr in Ihrem Computer.

Vielleicht hat Ihr PC überhaupt keine eingebaute Uhr oder sie arbeitet nicht mehr richtig. Auf jeden Fall müssen Sie, wollen Sie die richtige Zeit haben, jedesmal beim Starten umständlich die Systemzeit neu setzen. Meistens unterbleibt dieser unangenehme Vorgang jedoch – dann ist es schwierig festzustellen, wann man diesen oder jenen Brief, dies oder das Programm erarbeitet hat.

Damit nun die Chronologie der Ereignisse an Ihrem PC gewahrt bleibt, aber auch, um eine Kontrolle darüber zu haben, wann – vielleicht von einem Unbefugten – Ihr Computer zuletzt benutzt wurde, sollten Sie das Programm »rtimer.com« (Listing 1) verwenden. Besonders wenn man sich ab und an kleine Assembler-Programme schreibt – zahlreiche Abstürze des Computers müssen einkalkuliert werden – ist es sehr unangenehm, jedesmal die Zeit neu zu setzen, damit man noch unterscheiden kann, wann welche Version des Programms entstanden ist. Eine ganz andere Aufgabe kann dies Programm aber dann übernehmen, wenn mit der Hardware alles in Ordnung ist. Ohne jeden Aufwand können Sie damit eine Kontrolle darüber erhalten, wie lange Sie im Laufe der Zeit an Ihrem PC saßen, wann Sie »eingeloggt« waren. Das Programm erzeugt völlig selbständig eine Datei, in der die Zeiten festgehalten werden, in denen der Computer eingeschaltet war. Sie müssen sich weder beim Einschalten, noch beim Ausschalten darum kümmern. Auf diese Fähigkeit wird später genauer eingegangen.

Um diese Ziele zu verwirklichen, muß »rtimer« die Systemzeit und das Datum irgendwo speichern, und zwar so, daß die Werte auch nach dem Ausschalten des Computers erhalten bleiben. Da müssen natürlich die Speichermedien Festplatte oder auch Diskette herhalten; die Werte müssen also in einer Datei abgelegt werden.

Außerdem muß die Zeit in regelmäßigen Abständen aktualisiert werden, damit sie möglichst »frisch« zur Verfügung steht. Da bietet sich ein TSR-Programm (also ein Programm, das nach dem Aufruf im Speicher aktiv bleibt: »Terminate and Stay Resident«) an. Wie Sie aber sicher wissen, hat das Betriebssystem die unangenehme Eigenschaft, nicht »reentrant« zu sein. Dies bedeutet, daß DOS-Funktionen nicht durch andere DOS-Funktionen unterbrochen werden dürfen, weil DOS nicht »wiederbetretbar« ist.

Ein TSR-Programm hat aber gerade die Aufgabe, andere Programme entweder auf Tastendruck oder, wie in diesem Fall, zeitgesteuert zu beliebigen Zeitpunkten zu unterbrechen. Deshalb gilt in der Regel, daß ein TSR-Programm niemals DOS-Funktionen aufrufen darf. Und da haben Sie schon das Problem – »rtimer« soll ein residentes Programm werden, das zu beliebigen Zeiten das laufende Programm unterbricht, um auf eine Datei zuzugreifen und dort die Systemzeit zu speichern. Die Dateiverwaltung aber ist eine ureigene Angelegenheit des Betriebssystems. Wie kann man nun feststellen, ob gerade eine DOS-Funktion aktiv ist und das Programm demnach nicht ausgeführt werden darf? Oder anders gefragt: Wie kann man feststellen, daß DOS gerade keine Service-Programme ausführt und man Betriebssystemfunktionen aufrufen darf? Da bietet DOS selbst mehrere Lösungswege an, einer davon wird hier beschritten.

Jedesmal wenn das Betriebssystem nichts zu tun hat, und das ist erstaunlicherweise fast ständig der Fall, zum Beispiel wenn es auf einen Tastendruck wartet und sonst keine weitere Aufgabe ansteht, wird dies durch einen Aufruf der Interrupt-Service-Routine (ISR) des Interrupts 28hex angezeigt. Der Vektor zeigt standardmäßig einfach auf einen Iret-Befehl. Die Originalroutine macht also nichts anderes, als einen »Return-From-Interrupt« auszuführen.

Viele Wege führen zum Interrupt 28hex

Mit einem TSR-Programm kann man sich aber nun in diese Interrupt-Kette einhängen, aus diesem Grund wurde sie auch geschaffen. Alle MS-DOS-Aufrufe können dann innerhalb dieser ISR ohne Bedenken und die Gefahr eines Absturzes ausgeführt werden. Ein Programm kann sich innerhalb der Interrupt28h-Routine sogar selbst deinstallieren, wofür wiederum Betriebssystemfunktionen nötig sind, damit die Speicherverwaltung nicht durcheinander kommt.

Zur Dateibearbeitung kann man entweder die veralteten FCB-Funktionen benutzen oder, viel bequemer, die »neuen« Handle-Funktionen. Dabei übernimmt MS-DOS die komplette Verwaltung aller Zeiger und Werte, die mit dieser Datei zu tun haben. Sie sind mit diesen Funktionen sozusagen der Filmproduzent, der dem Regisseur sagt, wie er die Puppen (Dateien) tanzen lassen soll. Der Kleinkram (Zeiger verwalten, Informationen über die Datei ermitteln) braucht Sie dabei nicht zu beschäftigen. Darum kümmern sich »dienstbare Geister«.

Sie müssen zunächst nur festlegen, um welche Datei es gehen soll, später dann, was mit dieser Datei geschehen soll. Alle benötigten Informationen liefert Ihnen (vielmehr: Ihrem Programm) DOS dann in Registern zurück und stellt sie Ihrem Programm zur Verfügung. Um die Sache noch weiter zu vereinfachen, muß nicht jedesmal der Dateiname ange-

geben werden, auf den sich die folgende Dateifunktion beziehen soll, sondern es genügt, dem Betriebssystem eine Nummer zu übergeben, die dieser Datei beim Öffnen des Handles zugeteilt wurde. Allerdings dürfen Sie nicht vergessen, den Handle für eine Datei auch wieder zu schließen, da MS-DOS die Verwaltung übernimmt und dafür auch intern Speicherplatz belegt. Nur dann weiß das Betriebssystem, daß es den internen Speicherbereich wieder freigeben kann und mit Werten einer anderen Datei neu belegen darf.

Übrigens: Mit der Files-Angabe in der Datei »config.sys« benennen Sie genau die Anzahl an Handles, die MS-DOS gleichzeitig offen halten kann, für die es also beim Starten des Systems internen Speicherplatz reserviert hat. Je Handle sind dies etwa 60 Byte.

Wenn der reservierte Platz belegt ist, da zu viele Handles gleichzeitig offen gehalten werden sollen, das heißt, für zu viele Dateien Zeiger und Werte gespeichert werden sollen, dann ergibt dies eine Fehlermeldung.

Trickreich: Speichern ohne Platz zu belegen

Beim Beenden eines Programms mit der Funktion 4Chex übernimmt DOS allerdings sogar das Schließen aller offenen Dateien und damit die Freigabe des internen Speicherbereichs selbständig. Das Schließen des Handles bewirkt aber auch, daß beispielsweise das Datum der Dateierzeugung aktualisiert wird, wenn in die Datei geschrieben wurde. Hieran erkennen Sie schon, wie leistungsfähig die Handle-Funktionen sind: DOS verwaltet auch das Flag, das ohne Ihr Zutun das Schreiben in eine Datei anzeigt.

Um nun auf »rtimer« zurückzukommen: Die Funktion »Create File« erzeugt eine neue Datei, eine existierende wird dabei überschrieben, es sei denn, die Datei wäre schreibgeschützt. Es wird also kaum geprüft, ob die Datei schon auf der Festplatte oder der Diskette existiert. Die Dateilänge wird auf Null gesetzt, die Datei hat also keinen Inhalt. Das ist in unserem Fall nur von Vorteil: Eine Datei ohne Inhalt belegt nur wenig Speicherplatz im Directory, außerdem muß kein »Inhalt« auf das Speichermedium geschrieben werden.

Die Datei besteht vielmehr nur aus einem Eintrag im Verzeichnis, und dies wiederum belegt standardmäßig einen festgelegten Platz, unabhängig von der Anzahl der Einträge. Wenn ins Hauptverzeichnis geschrieben wird, ist sogar die Stelle auf der Diskette oder Festplatte festgelegt, nämlich ganz vorne (außen) auf dem Medium.

Wie kann man aber nun aus einer »leeren« Datei eine Zeit und ein Datum lesen? Ganz einfach: Beim Schließen eines Handles werden automatisch, wenn die Datei geändert wurde, Zeit- und Datumswerte aktualisiert.

Man kann also einfach diese beiden Werte verwenden, um daraus neue Systemwerte zu setzen.

Die Dateiwerte sind in einem speziellen Format gespeichert. Die Auswertung und die Umwandlung für das System geht aus Listing 2 »rtimer.asm« hervor.

Nun ist alles ganz einfach: Zunächst das Programm resident installieren, dann mit dem Timer-Interrupt, der etwa alle 18 Millisekunden aufgerufen wird, einen Zähler herabzählen. Wenn der Zähler bei 0 angelangt ist, wird ein Flag gesetzt, das der Routine des Interrupts 28hex anzeigt, daß es wieder an der Zeit ist, eine Leerdatei zu erzeugen.

Beim nächsten Starten werden dann die Dateiwerte für das Setzen der Systemwerte verwendet. Zugegeben: War der Computer länger ausgeschaltet, dann ist man nicht ganz auf dem neuesten Stand. Aber aktueller geht es ohne Echtzeituhr und ohne manuelle Eingriffe wirklich nicht. Vor allem sind Sie jetzt wirklich fast im »Laufe der Zeit«. Die Chronologie der Ereignisse wird festgehalten. Jeder Reset setzt sogar so gut wie die richtige Zeit. Außerdem brauchen nur einmal während einer Sitzung die Systemwerte von Hand gesetzt werden, damit das Ergebnis sofort übernommen wird. Folgende Zeile sollten Sie in Ihrer »autoexec.bat« angeben, wenn Sie ohne Echtzeituhr fortlaufende Zeiten wünschen:

```
rtimer /r
```

Fast genauso verhält sich das Programm in seiner Funktion als Sammler für die Arbeitszeiten am PC. Abgesehen vom Setzen der Systemwerte beim Starten ist die Arbeitsweise identisch. Wieder wird »rtimer« gleich in der »autoexec.bat« installiert. Auch die Dateiwerte werden in regelmäßigen Abständen auf den Stand der Systemwerte für Zeit und Datum gebracht. Diesmal allerdings interessieren die Dateiwerte im Hinblick darauf, wann die Arbeit am Computer beendet wurde. Dieser alte Wert wird zusammen mit der neuen Systemzeit einfach durch Ausgabeumleitung in eine Datei geschrieben. Man kann damit genau feststellen, ob, wenn ja, wann, und vor allem, wie lange der Computer benutzt wurde. Folgende zwei Zeilen, eingefügt in die Datei »autoexec.bat«, sind schon alles, was dazu erforderlich ist:

```
rtimer /f /n » login.dat
rtimer /n /5 » login.dat
```

(der Standardwert für das Aktualisieren »/1« entspricht einer Minute).

Zum Schluß noch einige Anmerkungen zum Programm-Listing. Viele Zeilen und Anweisungen in einem Programm sind sogenannte Assembler-Anweisungen, die entweder gar keinen eigenen Code erzeugen, oder sich nur, meist unbemerkt, im Code auswirken. Vielfach kennt man auch nicht die genaue Bedeutung und schreibt einfach immer wieder die Standardangaben hin. Einige dieser Direktiven werden jedoch im Programm bewußt eingesetzt und sollen deshalb etwas näher beleuchtet werden. Da ist zum einen die berüchtigte Assume-Anweisung (»Assume« heißt annehmen, vermuten). Der Assembler macht aus dieser Anweisung genau das, was die Übersetzung ausdrückt. Er »vermutet«, geht also davon aus, daß sich alle Speicheradressen auf das oder die in der Assume-Anweisung angegebenen Register beziehen. Das Standardregister des Mikroprozessors ist das DS-Register; ohne besondere Angaben bildet also die CPU zur Laufzeit eines Programms alle Speicheradreßbezüge aus DS und der angegebenen Speicherstelle. Durch den sogenannten Segment-Override-Befehl kann man aber diesen Standardwert überschreiben. Dieser Override ist Teil des Befehls, und dafür muß vom Assembler auch Code erzeugt werden. Meistens wird dieser Befehl auch tatsächlich ins Programm geschrieben, etwa in der Form:

```
mov cs:flag,0
```

Treten jedoch diese CS- oder ES-Anweisungen gehäuft auf, wie das in TSR-Programmen ja besonders der Fall ist, kann man sich die Sache einfacher machen, indem man ein Register spezifiziert, wie dies in Listing 2 zu sehen ist. Alle Adressen erhalten dann automatisch diesen Override-Befehl. Die nächste Assume-Anweisung hebt die Wirkung dann wieder auf, ersetzt oder ergänzt sie.

Bitte sehen Sie sich die kleine Mov-Zeile noch einmal an. Sie waren vielleicht schon einmal erstaunt, daß es beim Assemblieren manchmal bei einer solchen Zeile klappte, manchmal jedoch nicht. Die Lösung ist ganz einfach: Woher kann der Assembler wissen, ob er für diese Zeile den Code für den Befehl

```
mov cs:word ptr [flag],0000
```

oder den Code für den Befehl

```
mov cs:byte ptr [flag],00
```

erzeugen soll? Diese Information kann er nur dem Listing selbst entnehmen. Deshalb kann es zu Problemen kommen, wenn der Assembler bei der Bearbeitung dieses Befehls das Format von Flag noch nicht kennt.

(Jochen Bleichroth/bj)

```
[4S3]    1: ; Programm: RTIMER.ASM
[NU6]    2: ; Funktion: Vollständig automatisches
[PD3]    3: ;            Zeitprotokoll
[5Y4]    4: ; Sprache:  MASM ab Version 4.Ø
[5G4]    5: ; Autor:    Jochen Bleichroth
[ØØØ]    6:
[EH3]    7: code         segment byte
[4R4]    8: assume       cs:code,ds:nothing
[B55]    9:          ; Durch die Angabe 'DS: nothing'
[TS5]   1Ø:          ; weiß der Assembler, daß er alle
[CU5]   11:          ; folgenden Adressen auf CS: bezie-
[P35]   12:          ; hen soll ('DS:...' kann auch ein-
[4P5]   13:          ; fach weggelassen werden), er fügt
[K35]   14:          ; dann vor jeder Adresse den Seg-
[4Ø4]   15:          ; ment Override 'CS:' ein!
[ØØØ]   16:
[WC1]   17:             org   1ØØh
[ØØØ]   18:
[EG3]   19: delay_time  equ   444h
[7V5]   2Ø:          ; Delay-Zeit für den IntØ8 Counter
[8T4]   21:          ; (444h=1 Min., 2aaah= 1Ø Minuten)
[ØØØ]   22:
[ØV2]   23: msi         macro txt
[WD3]   24:             mov   si,offset txt
[HPØ]   25:             endm
[385]   26:          ; Dies kleine Macro wird weiter
[5B4]   27:          ; unten häufig gebraucht.
[ØØØ]   28:
[F64]   29: ;-------------------------------------
[2U3]   3Ø: resident:   jmp   start
[UF5]   31:          ; Der Sprung über den residenten
[Y62]   32:          ; Teil hinweg.
[ØE4]   33: orgintØ8    equ   this  dword
[FB5]   34:          ; Einige Werte, die im residenten
[S24]   35:          ; Teil gespeichert werden
[5L5]   36:          ; Zuerst der alte Vector des INTØ8
[HM2]   37: intØ8ofs    dw    Ø
[DM2]   38: intØ8seg    dw    Ø
[ØØØ]   39:
[8E4]   4Ø: orgint28    equ   this  dword
[4F4]   41:          ; Dann der Zeiger auf die alte
[X93]   42:          ; Rountine des Int28h
[LM2]   43: int28ofs    dw    Ø
[FM2]   44: int28seg    dw    Ø
[ØØØ]   45:
[5X3]   46: fname       db    "c:\rtime",Ø
[LW3]   47:          ; Der Default-Dateiname:
[BS4]   48:          ; Er endet mit einem Null-Byte
[PP4]   49: fname_ptr   dw    offset fname
[4G5]   5Ø:          ; Der Offset-Zeiger auf den Namen
[ØØØ]   51:
[GB4]   52: counter     dw    delay_time
[BW4]   53:          ; Dies ist der Zähler der IntØ8
[3W2]   54: count_flag  db    Ø
[UM5]   55:          ; Wenn der Zähler bei Null angekom-
[985]   56:          ; men ist, wird das Flag gesetzt.
[K35]   57:          ; Der Int28h fragt dieses Flag ab.
[F64]   58: ;-------------------------------------
[LP2]   59: newintØ8:   pushf
[B22]   6Ø:             dec   counter
[XZ4]   61:          ; Die INTØ8-Interrupt-Service-
[AP5]   62:          ; Routine (ISR), zählt den Counter
[EK3]   63:          ; 'runter bis auf Null.
[592]   64:             jnz   oldintØ8
[ØØØ]   65:
[3Z2]   66:             mov   count_flag,1
[L55]   67:          ; Dann setzt sie das Flag und den
[V46]   68:          ; Zähler wieder auf den Start-Wert.
[U54]   69:             mov   counter,delay_time
[S53]   7Ø: count       label word
[T76]   71:          ; Der Delay-Parameter ändert diesen
[W55]   72:          ; Start-Wert, deshalb das Label.
[KF2]   73: oldintØ8:   popf
[682]   74:             jmp   orgintØ8
[8R3]   75:          ; Sprung zur alten ISR
[F64]   76: ;-------------------------------------
[GM3]   77:          ; Die neue ISR des INT28:
[AP4]   78: newint28:   cmp   count_flag,Ø
[4L4]   79:          ; Vergleicht das Count-Flag.
[612]   8Ø:             je    oldint28
[T25]   81:          ; Wenn noch nicht gesetzt, dann
[ØX3]   82:          ; Sprung zur alten ISR.
[F91]   83:             push  ax
[L91]   84:             push  bx
[8B5]   85:          ; Wenn aber gesetzt, dann pushen
[7K4]   86:          ; der benötigten Register.
[R91]   87:             push  cx
[V91]   88:             push  dx
[P91]   89:             push  ds
[ØØØ]   9Ø:
[K91]   91:             push  cs
[DG4]   92:          ; DS auf CS setzen, weil der
[YH4]   93:          ; Zeiger auf den Dateinamen
[3Ø5]   94:          ; für die DOS-Datei-Funktionen
[AR3]   95:          ; in DS:DX stehen muß.
[G71]   96:             pop   ds
[RK3]   97:             mov   dx,ds:fname_ptr
[GT1]   98:             mov   ah,3ch
[1W3]   99:          ; Funktion: Create File
[AW1]  1ØØ:             xor   cx,cx
```

```
[UE5]  1Ø1:          ; Eine existierende Datei wird ra-
[HL5]  1Ø2:          ; dikal überschrieben und die Län-
[X35]  1Ø3:          ; ge der Datei auf Null gesetzt.
[NM4]  1Ø4:          ; (im Fehlerfall: Ende der ISR)
[N81]  1Ø5:             int   21h
[YV1]  1Ø6:             jc    ready28
[UV1]  1Ø7:             mov   bx,ax
[A14]  1Ø8:          ; Datei-Handle in BX laden
[ST1]  1Ø9:             mov   ah,3eh
[M36]  11Ø:          ;Erst das Dateischließen setzt das
[W16]  111:          ;aktuelle Datum und die Systemzeit
[N81]  112:             int   21h
[ØØØ]  113:
[A45]  114: ready28:    mov   ds:count_flag,Ø
[ZR3]  115:          ; Count-Falg löschen.
[E33]  116:          ; Register 'Popen'.
[G71]  117:             pop   ds
[N71]  118:             pop   dx
[H71]  119:             pop   cx
[Y25]  12Ø:          ; (Die Datei hat die Länge Null,
[E65]  121:          ; dh. die Datei besteht nur aus
[NW4]  122:          ; einem Eintrag im Directory!)
[D71]  123:             pop   bx
[971]  124:             pop   ax
[ØØØ]  125:
[HZ3]  126: oldint28:   jmp   orgint28
[VY4]  127:          ; Sprung zur alten Interrupt-
[7U3]  128:          ; Service-Routine (ISR)
[ØØØ]  129:
[F64]  13Ø: ;-------------------------------------
[9S4]  131:          ; Dies ist der transiente Teil
[955]  132:          ; des Programm, d.h. dieser Teil
[294]  133:          ; bleibt nicht resident im
[BN1]  134:          ; Speicher
[Ø23]  135: assume      ds:   code
[385]  136:          ; Ab hier sollen nun alle Adress-
[WZ4]  137:          ; Angaben auf DS: bezogen werden
[ØØØ]  138:
[LF3]  139: transient   label byte
[365]  14Ø:          ; Label für die Berechnung der
[X65]  141:          ; Programm-Länge des TSR-Teils
[M25]  142:          ; Diese 4 Zeilen sind Anweisun-
[LL4]  143:          ; gen für den Assembler. KEEP
[D45]  144:          ; verlangt die Länge in Para's.
[DG6]  145: isr_length  equ   (transient - resident)
[ZH6]  146: tsr_length  equ   (isr_length + 1ØØh + Øfh)
[S95]  147: tsr_para    equ   tsr_length /1Øh
[ØØØ]  148:
[BE5]  149:          ; Einige Werte, die in diesem Teil
[E55]  15Ø:          ; gespeichert werden. Durch die
[TE4]  151:          ; Angabe dieser Werte vor (!)
[695]  152:          ; diesem Programm-Teil spart man
[SØ6]  153:          ; die umständlichen Format-Angaben
[TL3]  154:          ; im Listing ,also z.B.:
[PN5]  155:          ;  'cmp ds:byte ptr [path_flag],Ø'
[GF5]  156:          ; Der Assembler kennt ja schon das
[RX3]  157:          ; Format (Byte oder Word)
[KX1]  158:          ; der Werte.
[9R2]  159: sav_time    dw    Ø
[DP2]  16Ø: sav_date    dw    Ø
[913]  161: instal_flag db    Ø
[AM2]  162: bad_param   db    Ø
[SP2]  163: path_flag   db    Ø
[NP2]  164: parm_flag   db    Ø
[5R2]  165: resi_flag   db    Ø
[NN2]  166: name_flag   db    Ø
[ER2]  167: show_flag   db    Ø
[UP2]  168: txt_flag    db    "&"
[ØØØ]  169:
[6B2]  17Ø: second      db    Ø
[7D2]  171: minute      db    Ø
[H72]  172: hour        db    Ø
[X42]  173: day         db    Ø
[H92]  174: month       db    Ø
[472]  175: year        dw    Ø
[ØØØ]  176:
[EV1]  177:          ; Die Texte
[DØ6]  178:          ; Durch die Ausgabe der Texte über
[BY4]  179:          ; die Funktion Ø2h des INT21h und
[A15]  18Ø:          ; nicht, wie üblich, über die
[DC5]  181:          ; Funktion Ø9h, ergeben sich zwei
[9Y1]  182:          ; Vorteile:
[E25]  183:          ; 1. Der Text steht ohne Unter-
[F24]  184:          ;    brechung durch '$' oder
[CE4]  185:          ;    Zeilenumbrüche als ge-
[ZD5]  186:          ;    schlossener Block im Programm
[VF4]  187:          ; 2. Die Texte können recht
[MM4]  188:          ;    flexibel miteinander kom-
[513]  189:          ;    biniert werden
[944]  19Ø: start_txt   db    "RTIMER ",Ø
[8M3]  191: ver_txt     db    "v1.Ø - ",Ø
[K26]  192: info_txt    db    "(c) 1991 DOS Internati"
[K24]  193:             db    "onal, Autor: Jochen "
[KH2]  194:             db    "Bleichroth"
[RH1]  195:             db    Ødh,Øah
[V53]  196:             db    "- Ungültiger "
[PR2]  197:             db    "Parameter -",Ø
[ØØØ]  198:
[1P6]  199: inst_txt    db    "Programm installiert",Ø
[YD4]  2ØØ: opt_txt     db    "Optionen",Ø
```

```
[5B4]  201: usage_txt   db    "] [[Pfad]"
[DS2]  202:             db    "Dateiname]",Ø
[785]  203: usage2_txt  db    ": S=Set SysT/D  "
[VK2]  204:             db    "R=/S&Inst.  "
[X83]  205:             db    "F=Show FileT/D  "
[RL2]  206:             db    "N=No Text  "
[633]  207:             db    "nn=Delay 1..1Ø",Ø
[ØØØ]  208:
[4K5]  209: path_txt    db    "Bei Installation "
[CT3]  210:             db    "Pfad mit angeben",Ø
[ØØØ]  211:
[MX4]  212:           ; Das wird besonders in diesen
[ZV3]  213:           ; Text-Teilen deutlich.
[RM4]  214:           ; Jeder Text wird, wie bei der
[7L5]  215:           ; Programmiersprache C, mit einem
[YD4]  216:           ; Zero-Byte abgeschlossen.
[ØØØ]  217:
[655]  218: datei_txt   db    " (Datei-Werte)",Ø
[FX4]  219: error_txt   db    "Fehler beim ",Ø
[XB4]  220: write_txt   db    "Schreib-",Ø
[HR4]  221: file_txt    db    "Zugriff auf: "
[563]  222: f_end_txt   db    "'",Ø
[WS4]  223: setz_txt    db    "Setzen von ",Ø
[G94]  224: system_txt  db    "System-"
[RB5]  225: time_txt    db    "Datum und Zeit",Ø
[DW4]  226: gsetzt_txt  db    " gesetzt ",Ø
[ØØØ]  227:
[F64]  228: ;-------------------------------------
[FX3]  229:           ; Diese Zeilen sind nur
[L95]  230:           ; Anweisungen für den Assembler.
[VK4]  231:           ; Es wird daraus kein Code er-
[DR4]  232:           ; zeugt. Sinn dieser Zeilen:
[RD5]  233:           ; Berechnung der Länge des Namen-
[U15]  234:           ; Textes, um damit den Schalter
[5M3]  235:           ; 'ungerade' auf 1 oder Ø
[XØ2]  236:           ; zu setzen.
[ØØØ]  237:
[ZD6]  238: txt_len     equ   info_txt-start_txt-1
[MF4]  239: tst_len     equ   txt_len - 1
[ND4]  240: cmp_len     equ   tst_len / 2
[7R4]  241: dmy_length  equ   txt_len / 2
[W36]  242: ungerade    equ   dmy_length - cmp_len
[ØØØ]  243:
[F64]  244: ;-------------------------------------
[ØØØ]  245:
[L54]  246: start:      call  parameter
[2Y5]  247:         ; Hier geht es los: Hierhin springt
[ZH5]  248:         ; das Programm zuerst. Das ist nach
[EF5]  249:         ; dem Auswerten der Parameter die
[AU5]  250:         ; 'Verteil-Zentrale'. Entweder Zeit
[XE5]  251:         ; und Datum setzen oder gleich re-
[GP5]  252:         ; sident installieren oder beenden
[HF5]  253:         ; aufgrund eines Parameterfehlers
[AR4]  254:         ; oder Zeigen der Datei-Daten.
[GW2]  255:           cmp   bad_param,-1
[BW2]  256:           jne   params_okay
[GV2]  257:           jmp   parm_error
[R95]  258: params_okay:cmp   show_flag,-1
[6L2]  259:           je    show_f_td
[963]  260:           cmp   instal_flag,-1
[ZX2]  261:           jne   try_install
[8T2]  262:           jmp   set_timdat
[F64]  263: ;-------------------------------------
[ØØØ]  264:
[9T4]  265: show_f_td:  call  get_timdat
[3K4]  266:           ; Also Zeigen der Datei-Daten
[1L2]  267:             jc    read_error
[BB5]  268:           ; Holen der Werte, bei Fehler Ende
[UW2]  269:             mov   sav_time,cx
[ZK3]  270:           ; FileTime steht in CX
[1V2]  271:             mov   sav_date,dx
[7H3]  272:           ; FileDate steht in DX
[WM2]  273:             call  make_time
[2Ø4]  274:           ; Zeit und Datums-Ausgabe
[372]  275:           ; vorbereiten
[HM2]  276:             call  make_date
[KE2]  277:             call  show_ftd
[ØZ2]  278:           ; Zeigen der Werte
[2P2]  279:             msi   datei_txt
[UZ2]  280:           ; Datei-Text laden
[ND1]  281:             call  txt
[7L2]  282:           ; Text ausgeben
[DN2]  283:             jmp   ende_prog
[EP2]  284:           ; ....dann Schluss
[F64]  285: ;-------------------------------------
[ØØØ]  286:
[Ø45]  287: try_install:cmp   path_flag,Ø
[4U4]  288:           ; Also Installations-Wunsch.
[CH2]  289:             je    path_okay
[3N3]  290:           ; Path muß komplett
[MP2]  291:           ; angegeben sein.
[5V2]  292:             jmp   path_error
[553]  293:           ; Wenn nicht: Fehler
[AL5]  294: path_okay:  mov   dx,ds:fname_ptr
[8A5]  295:           ; Zeiger auf Dateinamen in DS:DX.
[GT1]  296:             mov   ah,3ch
[FH5]  297:           ; Dann: Versuch, Datei zu erzeugen
[AW1]  298:             xor   cx,cx
[ZB5]  299:           ; Dateiattribut in CX (Ø = normale
[B71]  300:           ; Datei)
```

```
[N81]  301:             int   21h
[WP1]  302:             jc    error
[RF4]  303:           ; Fehler können z.B. sein:
[FG3]  304:           ; schreib-geschützt,
[Ø24]  305:           ; Path nicht gefunden etc.
[UV1]  306:             mov   bx,ax
[413]  307:           ; File-Handle in BX
[ST1]  308:             mov   ah,3eh
[546]  309:           ; Datei schließen (Dos schließt
[9P4]  310:           ; den angelegten Datei-Handle)
[N81]  311:             int   21h
[WP1]  312:             jc    error
[ZS3]  313:           ; Wenn Fehler ,dann Ende
[ØØØ]  314:
[2R2]  315:             call  get_timdat
[RW4]  316:           ; Abfrage Datei-Zeit und Datum
[MN2]  317:             jnc   do_install
[U85]  318:           ; Wenn alles okay: installieren
[AX4]  319: read_error: jmp   file_error
[6D3]  320:           ; Sonst Datei-Fehler
[EH4]  321: error:      jmp   write_error
[UK4]  322:           ; oder Datei-Schreib-Fehler.
[F64]  323: ;-------------------------------------
[Y35]  324: do_install: mov   sav_time,cx
[ZK3]  325:           ; FileTime steht in CX
[1V2]  326:             mov   sav_date,dx
[7H3]  327:           ; FileDate steht in DX
[WM2]  328:             call  make_time
[2Ø4]  329:           ; Zeit und Datums-Ausgabe
[372]  330:           ; vorbereiten
[HM2]  331:             call  make_date
[ØØØ]  332:
[LC5]  333:           ; Nun kann installiert werden mit
[RG5]  334:           ; GetInterruptVector des INT21h:
[S12]  335:             mov   ax,35Ø8h
[N81]  336:             int   21h
[ZM2]  337:             mov   intØ8ofs,bx
[9B5]  338:           ; Zeiger steht in ES:BX und wird
[P84]  339:           ; im TSR-Teil gespeichert.
[PL2]  340:             mov   intØ8seg,es
[N12]  341:             mov   ax,25Ø8h
[ZA5]  342:           ; Jetzt SetInterruptVector der
[2C5]  343:           ; neuen Routine, der neue Zeiger
[BH5]  344:           ; muß in DS:DX übergeben werden.
[F14]  345:             mov   dx,offset newintØ8
[N81]  346:             int   21h
[U12]  347:             mov   ax,3528h
[H85]  348:           ; Dasselbe für den Interrupt 28h:
[N81]  349:             int   21h
[5N2]  350:             mov   int28ofs,bx
[WL2]  351:             mov   int28seg,es
[R12]  352:             mov   ax,2528h
[R14]  353:             mov   dx,offset newint28
[N81]  354:             int   21h
[DY2]  355:             cmp   resi_flag,-1
[ML4]  356:           ; Hier geht es nur noch um die
[D92]  357:             je    resi_txt
[PZ4]  358:           ; Ausgabe der 'richtigen' Texte:
[3E2]  359:             call  name_txt
[GY3]  360:             jmp   short txt_fertig
[WV4]  361: resi_txt:   mov   dl,txt_flag
[DR1]  362:             mov   dh,2Øh
[R82]  363:             call  two_txt
[CP4]  364: txt_fertig: msi   inst_txt
[ND1]  365:             call  txt
[YY2]  366:             call  show_timdat
[9P3]  367:           ; Zeige Zeit und Datum
[E82]  368:             call  carriage
[TK5]  369:           ; Für die Funktion KEEP des INT21h
[WR5]  370:           ; muß die Länge des TSR-Teils (DX)
[B45]  371:           ; in Paragraphen (1 Para=16 Byte)
[663]  372:           ; angegeben werden.
[ZV2]  373:             mov   dx,tsr_para
[SØ2]  374:             mov   ax,31ØØh
[N81]  375:             int   21h
[ØØØ]  376:
[F64]  377: ;-------------------------------------
[N15]  378: set_timdat: cmp   bad_param,-1
[D95]  379:           ; Also Setzen von Zeit und Datum:
[SM2]  380:             je    parm_error
[9K5]  381:           ; Wenn Parameter-Fehler dann Ende.
[ØØØ]  382:
[2R2]  383:             call  get_timdat
[RW4]  384:           ; Abfrage Datei-Zeit und Datum
[EK2]  385:             jc    file_error
[ØØØ]  386:
[HM2]  387:             call  make_date
[MX2]  388:           ; Datum umwandeln
[9T1]  389:             mov   ah,2bh
[V94]  390:           ; dann SystemDatum setzen.
[N81]  391:             int   21h
[5L1]  392:             cmp   al,ØØ
[DY4]  393:           ; Wenn okay, dann AL=ØØ sonst FF
[XT2]  394:             jne   time_error
[F64]  395: ;-------------------------------------
[WM2]  396:             call  make_time
[CN2]  397:           ; Zeit umwandeln
```

Listing 1. Mit »rtimer.com« können Sie auch ohne Echtzeituhr die Systemzeit setzen und vollautomatisch die Dauer von Arbeitssitzungen protokollieren

```
[HT1]  398:                mov   ah,2dh
[K64]  399:            ; dann SystemZeit setzen.
[N81]  4ØØ:                int   21h
[5L1]  4Ø1:                cmp   al,ØØ
[DY4]  4Ø2:            ; Wenn okay, dann AL=ØØ sonst FF
[XT2]  4Ø3:                jne   time_error
[ØØØ]  4Ø4:
[3E2]  4Ø5:                call  name_txt
[SY4]  4Ø6:            ; Alles in Ordnung, jetzt Text
[ML1]  4Ø7:            ; ausgeben
[DW2]  4Ø8:                msi   system_txt
[ND1]  4Ø9:                call  txt
[PW2]  41Ø:                msi   gsetzt_txt
[ND1]  411:                call  txt
[TS4]  412:            ; Nun prüfen, ob installiert
[B15]  413:            ; werden soll, wenn nicht: Ende
[DY2]  414:                cmp   resi_flag,-1
[4Ø2]  415:                jne   beenden
[LT4]  416:            ; ansonsten nun die gesamte
[M15]  417:            ; Install-Routine durchlaufen.
[8Z2]  418:                jmp   try_install
[WM4]  419: beenden:       call  show_timdat
[9P3]  42Ø:            ; Zeige Zeit und Datum
[FN3]  421:                jmp   short ende_prog
[FX2]  422:            ; Und dann Schluß
[F64]  423: ;-------------------------------------
[NP4]  424:            ; Es folgen die Routinen, die
[365]  425:            ; angesprungen werden, wenn ein
[Ø64]  426:            ; Fehler aufgetreten ist:
[9Ø5]  427: path_error: mov   txt_flag,"&"
[3E2]  428:                call  name_txt
[MF2]  429:                msi   path_txt
[BW2]  43Ø:                jmp   short usage
[6Ø5]  431: parm_error:  mov   txt_flag,"&"
[3E2]  432:                call  name_txt
[YE2]  433:                msi   info_txt
[X94]  434: Usage:         call  show_usage
[ZN3]  435:                jmp   short prog_ende
[165]  436: write_error:call  show_error
[ØØØ]  437:
[6T2]  438:                msi   write_txt
[8Ø5]  439:            ; Durch das Macro 'msi' wird die
[515]  44Ø:            ; Schreibarbeit in diesem Teil
[MZ4]  441:            ; erleichtert und das Programm
[XB3]  442:            ; übersichtlicher.
[ND1]  443:                call  txt
[NZ2]  444:                jmp   short go_on
[415]  445: file_error:  call  show_error
[B14]  446: go_on:         msi   file_txt
[AU4]  447:            ; Hier wird der Datei-Pointer
[ND1]  448:                call  txt
[RL4]  449:            ; geladen. Auch der Dateiname
[ØH3]  45Ø:                mov   si,[fname_ptr]
[FN4]  451:            ; endet mit einem Null-Byte.
[ND1]  452:                call  txt
[HP2]  453:                msi   f_end_txt
[Ø95]  454:            ; Alle Fehler-Routinen, aber auch
[PA5]  455:            ; die Get-/Set-Date/Time Routinen
[B15]  456:            ; (wenn also nicht installiert
[TE5]  457:            ; wird) springen letzten Endes zu
[RG5]  458:            ; der Marke 'Prog_ende'. Dort wird
[L35]  459:            ; noch ein Zeilenumbruch ausge-
[AØ5]  46Ø:            ; geben. Dann wird das Programm
[TV4]  461:            ; beendet, wobei der Exitcode 1
[KL4]  462:            ; gesetzt wird, der mit dem
[PU4]  463:            ; Errorlevel abgefragt werden
[141]  464:            ; kann.
[ZN3]  465:                jmp   short prog_ende
[425]  466: time_error:  call  show_error
[UH2]  467:                msi   setz_txt
[ND1]  468:                call  txt
[NF2]  469:                msi   time_txt
[FD3]  47Ø: prog_ende:   call  txt
[S74]  471: ende_prog:   call  carriage
[532]  472:                mov   ax,4cØ1h
[N81]  473:                int   21h
[F64]  474: ;-------------------------------------
[ØØØ]  475:
[PS3]  476: show_error   proc  near
[8W4]  477:            ; Aufgabe: Gibt den Namen-Text
[975]  478:            ; und einen allgemeinen Fehler-
[F15]  479:            ; Text aus, und zwar auch dann,
[755]  48Ø:            ; wenn die Text-Ausgabe eigent-
[LG5]  481:            ; lich unterdrückt wurde mit dem
[GK2]  482:            ; Parameter '/n'
[BU2]  483:                mov   txt_flag,"&"
[3E2]  484:                call  name_txt
[NS2]  485:                msi   error_txt
[ND1]  486:                call  txt
[3PØ]  487:                ret
[Y91]  488:            ; Fertig
[ØØØ]  489:
[ØU2]  49Ø: show_error   endp
[ØØØ]  491:
[F64]  492: ;-------------------------------------
[ØØØ]  493:
[FP3]  494: get_timdat   proc  near
[VZ4]  495:            ; Aufgabe: Fragt Datei-Zeit und
[4A2]  496:            ;          Datum ab
[432]  497:                mov   ax,3dØØh
```

```
[R15]  498:            ; Funktion: Öffnen einer Datei
[RP2]  499:            ;           via Handle
[AW1]  5ØØ:                xor   cx,cx
[525]  5Ø1:            ; (AL=Ø: Öffnen nur zum Lesen)
[X33]  5Ø2:                mov   dx,fname_ptr
[C75]  5Ø3:            ; Zeiger auf Dateinamen in DS:DX
[N81]  5Ø4:                int   21h
[FY1]  5Ø5:                jc    f_error
[TP2]  5Ø6:            ; Bei Fehler: Ende
[ØØØ]  5Ø7:
[UV1]  5Ø8:                mov   bx,ax
[A15]  5Ø9:            ; Handle-Nummer von AX nach BX
[P12]  51Ø:                mov   ax,57ØØh
[LS5]  511:            ; Funktion: Get/Set-File Time+Date
[N81]  512:                int   21h
[S85]  513:            ; mit der HandleNummer (AL=Ø: Get)
[FY1]  514:                jc    f_error
[TP2]  515:            ; Bei Fehler: Ende
[UW2]  516:                mov   sav_time,cx
[ZK3]  517:            ; FileTime steht in CX
[1V2]  518:                mov   sav_date,dx
[7H3]  519:            ; FileDate steht in DX
[ØØØ]  52Ø:
[ST1]  521:                mov   ah,3eh
[955]  522:            ; Jetzt den *Handle* schließen
[N81]  523:                int   21h
[TW4]  524:            ; (Die *Datei* müsste nicht ge-
[115]  525:            ; schlossen werden, da das Pro-
[M55]  526:            ; gramm keinen Inhalt schreibt.)
[FY1]  527:                jc    f_error
[5H2]  528: f_error:      ret
[TR2]  529: get_timdat   endp
[ØØØ]  53Ø:
[3C4]  531: ;-------------------------------------
[ØØØ]  532:
[GH3]  533: name_txt      proc  near
[2Ø5]  534:            ; Aufgabe: Ausgabe des Programm-
[ZØ2]  535:            ;          Namens.
[MS2]  536:                cmp   name_flag,Ø
[RE5]  537:            ; Dabei wird geprüft, ob der Name
[HC5]  538:            ; schon ausgegeben wurde. Wenn es
[515]  539:            ; der Fall ist - Ausgabe von ent-
[HC5]  54Ø:            ; sprechend vielen Leerzeichen in
[BB3]  541:            ; einer neuen Zeile.
[P72]  542:                je    name_out
[E82]  543:                call  carriage
[LC3]  544:                mov   si,dmy_length
[HW1]  545:                xor   dx,dx
[ZU3]  546: dummy:         call  two_txt
[Ø15]  547:            ; je zwei Leerzeichen ausgeben
[S51]  548:                dec   si
[S85]  549:            ; Solange bis SI gleich Null ist
[GZ1]  55Ø:                jnz   dummy
[162]  551:            ; Also nochmal
[F64]  552: ;-------------------------------------
[195]  553:            ; Wenn der Namen-Text eine ungera-
[6H5]  554:            ; de Anzahl Zeichen hat, weiteres
[WD5]  555:            ; Leerzeichen ausgeben. Wird beim
[T84]  556:            ; Assemblieren festgelegt!
[KV1]  557: if ungerade
[672]  558:                call  one_txt
[TUØ]  559: endif
[F64]  56Ø: ;-------------------------------------
[3PØ]  561:                ret
[EZ3]  562:            ; Entweder hier Fertig...
[KV4]  563: name_out:     mov   name_flag,-1
[5Z4]  564:            ; Also Offset Namen-Text in SI
[YT2]  565:                msi   start_txt
[UE5]  566:            ; laden und dann Textausgabe auf-
[M91]  567:            ; rufen.
[ND1]  568:                call  txt
[592]  569:                msi   ver_txt
[ND1]  57Ø:                call  txt
[SG2]  571: name_end:      ret
[Y93]  572:            ; ...oder hier Fertig.
[ØØØ]  573:
[UK2]  574: name_txt      endp
[F64]  575: ;-------------------------------------
[3C3]  576: carriage      proc  near
[ØØØ]  577:
[942]  578:                mov   dx,ØdØah
[CH5]  579:            ; Aufgabe: Ausgabe Carriage-Return
[6D2]  58Ø:            ;          ausgeben
[ØT1]  581:                xchg  dl,dh
[ØØØ]  582:
[KK3]  583: two_txt       proc  near
[1Ø5]  584:            ; Aufgabe: Zwei Zeichen (in DX)
[VV4]  585:            ;          ausgeben über Funktion
[1R2]  586:            ;          Ø2 des INT21h
[3R1]  587:                mov   ah,Ø2h
[N81]  588:                int   21h
[ØT1]  589:                xchg  dl,dh
[VY4]  59Ø:            ; jetzt zweites Zeichen in DL
[ØØØ]  591:
[5G3]  592: one_txt       proc  near
[6X4]  593:            ; Aufgabe: Ein Zeichen ausgeben
[YV2]  594:            ;          Zeichen in DL
[3V2]  595:            ;          über INT21h
[3R1]  596:                mov   ah,Ø2h
```

```
[N81]  597:              int   21h
[3PØ]  598:              ret
[Y91]  599:          ; Fertig
[ØØØ]  6ØØ:
[FH2]  6Ø1: one_txt      endp
[5G4]  6Ø2:          ; Alle drei Procs enden hier
[WL2]  6Ø3: two_txt      endp
[R74]  6Ø4:          ; Das Letzte endet zuerst!
[DD2]  6Ø5: carriage     endp
[F64]  6Ø6: ;-----------------------------------
[A23]  6Ø7: txt          proc  near
[P55]  6Ø8:          ; Aufgabe: Ausgabe von Text über
[7R3]  6Ø9:          ;          Funktion Ø2 Int21h
[TS2]  61Ø:              cmp   txt_flag,"&"
[492]  611:              je    txt_okay
[M74]  612:              cmp   si,offset time_txt
[952]  613:              jne   txt_end
[9H2]  614: txt_okay:    cld
[6L4]  615:          ; Der Text-Anfang steht in SI
[3R1]  616:              mov   ah,Ø2h
[WA5]  617:          ; Text endet mit einem Null-Byte!
[KR2]  618: get_txt:     lodsb
[W93]  619:          ; DS:SI nach AL laden
[FL1]  62Ø:              or    al,al
[WP3]  621:          ; Ist es das Null-Byte?
[T12]  622:              jz    txt_end
[PA4]  623:          ; Dann ist der Text zuende.
[6U1]  624:              mov   dl,al
[LØ5]  625:          ; Sonst Zeichen in DL schieben
[N81]  626:              int   21h
[ZS3]  627:          ; Und Zeichen ausgeben.
[ZG3]  628:              jmp   short get_txt
[ØX3]  629:          ; Nächstes Zeichen Laden
[ØH2]  63Ø: txt_end:     ret
[Y91]  631:          ; Fertig
[M32]  632: txt          endp
[F64]  633: ;-----------------------------------
[SH3]  634: parameter    proc  near
[4A5]  635:          ; Aufgabe: Auswertung der Komman-
[A42]  636:          ;          dozeile
[WT1]  637:              mov   si,8Øh
[2B5]  638:          ; Hier steht, wieviele Zeichen die
[UB4]  639:          ; Kommandozeile enthält.
[DP2]  64Ø:              mov   bl,ds:[si]
[PL1]  641:              or    bl,bl
[LN5]  642:          ; Test: Keine Parameter angegeben.
[XG2]  643:              jz    parms_end
[UØ2]  644:          ; Dann fertig
[LT1]  645:              xor   bh,bh
[T73]  646:          ; Sonst auswerten:
[4U1]  647:              mov   si,82h
[ØE5]  648:          ; Hier fängt die Kommandozeile an
[LR1]  649:              add   bx,si
[2S2]  65Ø:          ; Zeiger auf Ende
[ULØ]  651:              cld
[AD4]  652:          ; Richtungs-Flag löschen
[PL2]  653: get_end:     lodsb
[3N2]  654:          ; Byte in AL laden
[ØW1]  655:              cmp   si,bx
[SH2]  656:          ; Ende erreicht?
[5H2]  657:              ja    get_start
[GH3]  658:          ; Dann also Auswerten
[XG1]  659:              cmp   al," "
[BS5]  66Ø:          ; Leerzeichen in der Kommandozeile
[PR5]  661:          ; durch Null-Bytes ersetzen, damit
[LN5]  662:          ; ein eventuell angegebener Datei-
[315]  663:          ; Name garantiert mit Null abge-
[2B5]  664:          ; schlossen wird, wie es die DOS-
[9Ø6]  665:          ; Funktionen zur Dateiverarbeitung
[9Z4]  666:          ; verlangen. SI wurde ja auto-
[MF5]  667:          ; matischer erhöht, deshalb SI-1.
[5G2]  668:              je    make_null
[XR1]  669:              cmp   al,Ødh
[5G2]  67Ø:              je    make_null
[HR1]  671:              cmp   al,Øah
[232]  672:              jne   get_end
[ØH5]  673: make_null:   mov   byte ptr [si-1],Ø
[3D3]  674:              jmp   short get_end
[ØØØ]  675:
[YT3]  676: get_start:   mov   si,82h
[LW4]  677:          ; Kommandozeilen Anfang laden
[3G3]  678:              jmp   short get_ptr
[AX4]  679:          ; und zu Auswertung springen
[ØØØ]  68Ø:
[9N4]  681: parse_it:    call  pars_param
[YR3]  682:          ; Parameter bearbeiten
[LD5]  683:          ; Auswertung der Kommandozeile:
[PP2]  684: get_ptr:     lodsb
[PF4]  685:          ; Nächstes Byte in AL laden
[ØW1]  686:              cmp   si,bx
[SH2]  687:          ; Ende erreicht?
[KE2]  688:              ja    parms_end
[DL4]  689:          ; Dann Fertig mit Parametern
[5D1]  69Ø:              cmp   al,Ø
[PY1]  691:          ; Null-Byte?
[1Ø2]  692:              je    get_ptr
[UD4]  693:          ; Dann nächstes Byte Laden
[TK1]  694:              cmp   al,"/"
[TØ3]  695:          ; Schräg-Strich?
```

```
[262]  696:              je    parse_it
[5N4]  697:          ; Dann: Parameter bearbeiten
[EL2]  698:              call  get_fname
[256]  699:          ; Sonst muß es ein Datei-Name sein
[PN2]  7ØØ: Parms_end:   ret
[7S5]  7Ø1:          ; Alles Fertig, deshalb zurück zum
[BP1]  7Ø2:          ; Aufrufer
[2L2]  7Ø3: parameter    endp
[F64]  7Ø4: ;-----------------------------------
[NN3]  7Ø5: pars_param   proc  near
[XD5]  7Ø6:          ; Aufgabe: Auswertung eines Para-
[Ø25]  7Ø7:          ; meters, der mit '/' eingeleitet
[FN5]  7Ø8:          ; war /S ist mit andern Parametern
[L55]  7Ø9:          ; unverträglich, Wenn /S schon
[SB5]  71Ø:          ; ausgewertet, dann leider falsch
[BY2]  711:              cmp   parm_flag,-1
[3E2]  712:              je    bad_parse
[PWØ]  713:              lodsw
[1M1]  714:              cmp   al,"9"
[WS4]  715:          ; Sonst in AL und vergleichen
[9P1]  716:              ja    parse
[1L1]  717:              cmp   al,"1"
[6V4]  718:          ; Wenn keine Zahl, dann ist es
[CP1]  719:              jb    parse
[N72]  72Ø:          ; ein Zeichen?
[GV2]  721:              call  parse_delay
[UR2]  722:          ; Zahl auswerten
[3PØ]  723:              ret
[EZ3]  724:          ; Entweder hier Fertig...
[RX2]  725: parse:       cmp   ah,Ø
[2C5]  726:          ; AH muß Null sein, weil Option
[KL2]  727:              jne   bad_parse
[LU4]  728:          ; immer nur ein Zeichen lang
[7N2]  729:              call  get_param
[SA3]  73Ø:          ; Zeichen auswerten
[3PØ]  731:              ret
[4H3]  732:          ; ...oder hier Fertig...
[EU4]  733: bad_parse:   mov   bad_param,-1
[1A3]  734:          ; Fehler-Flag setzen
[3PØ]  735:              ret
[Y93]  736:          ; ...oder hier Fertig.
[ZP2]  737: pars_param   endp
[F64]  738: ;-----------------------------------
[WH3]  739: get_param    proc  near
[ØR5]  74Ø:          ;Aufgabe: Auswertung eines Zeichen
[865]  741:          ;als Param. und setzen von Flags
[UU1]  742:              and   al,Ødfh
[WM1]  743:          ; Upcase !!!
[9R1]  744:              cmp   al,"S"
[ND2]  745:          ; Ist's ein 'S'?
[742]  746:              jne   parse_r
[V73]  747:              mov   instal_flag,-1
[D34]  748:          ; Also nicht installieren
[XZ2]  749:              mov   parm_flag,-1
[GL5]  75Ø:          ; Keine anderen Parameter erlaubt
[3PØ]  751:              ret
[S34]  752:          ; Entweder hier Fertig....
[CK3]  753: parse_r:     cmp   al,"R"
[KD2]  754:          ; Ist's ein 'R'?
[F32]  755:              jne   parse_f
[ZZ2]  756:              mov   resi_flag,-1
[A44]  757:          ; Später resident machen
[V73]  758:              mov   instal_flag,-1
[V95]  759:          ; Aber nicht gleich installieren
[3PØ]  76Ø:              ret
[4H3]  761:          ; ...oder hier Fertig...
[CF3]  762: parse_f:     cmp   al,"F"
[EC2]  763:          ; Ist's ein 'F'?
[Z32]  764:              jne   parse_n
[813]  765:              mov   show_flag,-1
[GY3]  766:          ; Also Datei-Werte zeigen
[3PØ]  767:              ret
[4H3]  768:          ; ...oder hier Fertig...
[CH3]  769: parse_n:     cmp   al,"N"
[6D2]  77Ø:          ; Ist's ein 'N'?
[7L2]  771:              jne   parse_end
[K15]  772:          ; Nein: dann stimmt 'was nicht
[ZT2]  773:              mov   txt_flag," "
[AL4]  774:          ; Also Text *nicht* anzeigen
[3PØ]  775:              ret
[4H3]  776:          ; ...oder hier Fertig...
[ZW4]  777: parse_end:   mov   bad_param,-1
[4Y4]  778:          ; Leider schlechter Parameter
[3PØ]  779:              ret
[Y93]  78Ø:          ; ...oder hier Fertig.
[6L2]  781: get_param    endp
[F64]  782: ;-----------------------------------
[3W3]  783: parse_delay proc  near
[345]  784:          ; Aufgabe: Delay-Zeit auswerten
[9V4]  785:          ; und an die richtigen Stellen
[1N3]  786:          ; im Programm schieben
[1D1]  787:              cmp   ah,Ø
[XE2]  788:              je    get_delay
[U95]  789:          ; Wenn nur eine Ziffer, dann okay
[TK1]  79Ø:              cmp   ah,"Ø"
```

Listing 1. Mit »rtimer.com« können Sie auch ohne Echtzeituhr die Systemzeit setzen und vollautomatisch die Dauer von Arbeitssitzungen protokollieren

```
[1C5]  791:           ; Zweite Ziffer darf nur 'Ø' sein
[MK2]  792:             jne   bad_delay
[GG3]  793:           ; Sonst leider falsch
[1L1]  794:             cmp   al,"1"
[UG5]  795:           ; Aber erste Ziffer muß 1 sein
[MK2]  796:             jne   bad_delay
[GG3]  797:           ; Sonst leider falsch
[CVØ]  798:             lodsb
[T75]  799:           ; Danach muß Leerstelle folgen
[5D1]  8ØØ:             cmp   al,Ø
[V75]  8Ø1:           ; oder die Kommandozeile zuende
[MK2]  8Ø2:             je    next_okay
[3ZØ]  8Ø3:           ; sein
[3K3]  8Ø4:             jmp   short bad_delay
[GG3]  8Ø5:           ; Sonst leider falsch
[3V3]  8Ø6: next_okay:  mov   al,3ah
[4X2]  8Ø7:           ; also 'zehn' in AL
[2P3]  8Ø8: get_delay:  xor   al,3Øh
[265]  8Ø9:           ; Aus ASCII-Zahl eine Zahl machen
[CT1]  81Ø:             xor   ah,ah
[835]  811:           ; Multiplikation vorbereiten:
[HB3]  812:             mov   cx,delay_time
[KH4]  813:           ; Setzt der Assembler ein!!!
[D71]  814:             mul   cx
[DV2]  815:           ; Multiplizieren
[FK2]  816:             mov   count-2,ax
[FS4]  817:           ; Und in die Speicher-Stellen
[7N2]  818:             mov   counter,ax
[UM1]  819:           ; schieben
[3PØ]  82Ø:             ret
[EZ3]  821:           ; Entweder hier Fertig...
[ØU4]  822: bad_delay:  mov   bad_param,-1
[1A3]  823:           ; Fehler-Flag setzen
[3PØ]  824:             ret
[Y93]  825:           ; ...oder hier Fertig.
[DX2]  826: parse_delay endp
[F64]  827: ;-------------------------------------
[7G3]  828: get_fname   proc  near
[G84]  829:           ; Aufgabe: Datei-Name in der
[A15]  83Ø:           ;          Kommandozeile auswerten
[CVØ]  831:             lodsb
[G95]  832:           ; Prüfen, ob Pfad incl. Laufwerk
[3L3]  833:           ; mit angegeben wurde.
[S51]  834:             dec   si
[5M1]  835:             cmp   al,":"
[WW4]  836:           ; Zweites Zeichen ist dann ':'.
[1U1]  837:             je    get_it
[NY1]  838:           ; Alles Okay
[DY2]  839:             cmp   resi_flag,-1
[KZ4]  84Ø:           ; Pfad muß nur beim Resident-
[844]  841:           ; machen angegeben werden.
[ØY1]  842:             je    no_path
[963]  843:             cmp   instal_flag,-1
[1U1]  844:             je    get_it
[LM5]  845:           ; Wenn Resident und kein Pfadname,
[S32]  846:           ; dann Fehler.
[ZU4]  847: no_path:    mov   path_flag,-1
[UW2]  848: get_it:     dec   si
[ZV1]  849:           ; Also Okay:
[K33]  85Ø:             mov   fname_ptr,si
[GP5]  851:           ; Pointer auf Dateinamen speichern
[3PØ]  852:             ret
[UC1]  853:           ; Fertig.
[HH2]  854: get_fname   endp
[F64]  855: ;-------------------------------------
[4F3]  856: make_date   proc  near
[7V4]  857:           ; Aufgabe: Wandelt das, in einem
[XP5]  858:           ; speziellen Format gespeicherte,
[EA5]  859:           ; Datei-Datum in das Datum-Format
[3D5]  86Ø:           ; für die Funktion SetDate des INT
[W35]  861:           ; 21h um. Datum war in zwei Byte
[9G5]  862:           ; gespeichert: JJJJJJJM MMMTTTTT
[KS2]  863:             mov   ax,sav_date
[UV1]  864:             mov   bx,ax
[2W1]  865:             mov   dx,ax
[UF1]  866:             mov   cl,9
[F33]  867:           ; Bilde Jahr in BX:
[RT1]  868:             shr   bx,cl
[D72]  869:           ; Jahre ab 198Ø!
[5R1]  87Ø:             add   bx,198Ø
[8A5]  871:           ; Man muß also noch 198Ø addieren
[ØØØ]  872:
[RU1]  873:             and   ax,1eØh
[P63]  874:           ; Bilde Monat in AX:
[EF1]  875:             mov   cl,5
[LT1]  876:             shr   ax,cl
[ØØØ]  877:
[FR1]  878:             and   dx,1fh
[YH2]  879:           ; Bilde Tag in DL
[AG5]  88Ø:           ; Für SetDate muß Tag in DL sein,
[2U1]  881:             mov   dh,al
```

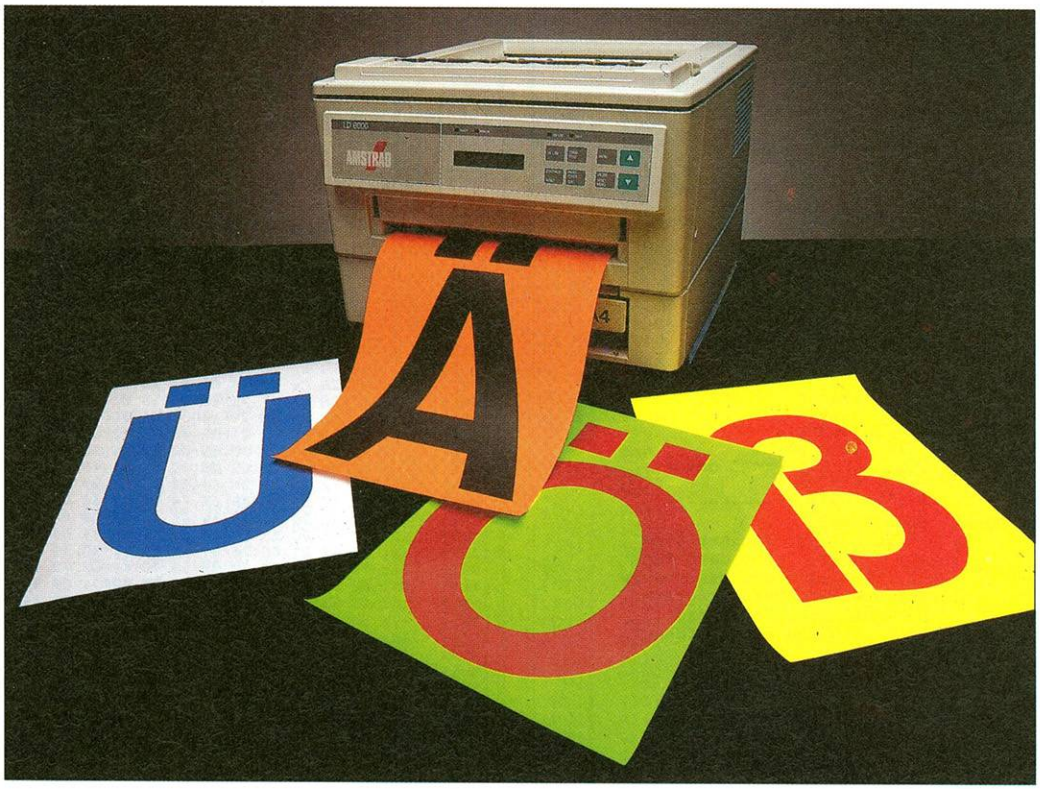
```
[NM3]  882:           ;             der Monat in DH,
[1W1]  883:             mov   cx,bx
[NN3]  884:           ;              und das Jahr  in CX
[2W1]  885:             mov   day,dl
[FH5]  886:           ; Alle drei Werte speichern, damit
[Z75]  887:           ; sie später ausgegeben werden
[3C1]  888:           ; können
[6B2]  889:             mov   month,dh
[332]  89Ø:             mov   year,cx
[3PØ]  891:             ret
[UC1]  892:           ; Fertig.
[EG2]  893: make_date   endp
[F64]  894: ;-------------------------------------
[DH3]  895: make_time   proc  near
[DU4]  896:           ; Aufgabe: Wandelt die, in einem
[XP5]  897:           ; speziellen Format gespeicherte,
[PB5]  898:           ; Datei-Zeit in das Zeit-Format
[XE5]  899:           ; für die Funktion SetTime des INT
[LS4]  9ØØ:           ; 21h um. Zeit war in zwei Byte
[RF5]  9Ø1:           ; gespeichert: HHHHHMMM MMMSSSSS
[VU2]  9Ø2:             mov   ax,sav_time
[UV1]  9Ø3:             mov   bx,ax
[2W1]  9Ø4:             mov   dx,ax
[6N1]  9Ø5:             mov   cl,11
[F73]  9Ø6:           ; Bilde Stunde in AX:
[LT1]  9Ø7:             shr   ax,cl
[H63]  9Ø8:           ; Bilde Minute in BX:
[DV1]  9Ø9:             and   bx,7eØh
[7S3]  91Ø:           ; Minuten ausmaskieren
[EF1]  911:             mov   cl,5
[FU3]  912:           ; und durch Schieben in
[RT1]  913:             shr   bx,cl
[MW1]  914:           ; BL bringen
[AV3]  915:           ; Bilde Sekunden in DX:
[FR1]  916:             and   dx,1fh
[YA2]  917:           ; Ausmaskieren
[AS1]  918:             add   dx,dx
[9F5]  919:           ; und verdoppeln, da im Datei-For-
[ØT1]  92Ø:             xchg  dl,dh
[TF5]  921:           ; mat Sekunden /2 gespeichert sind
[CS5]  922:           ; In Register schieben für SetTime
[YT1]  923:             mov   ch,al
[ØX4]  924:           ; Stunden in CH, Minuten in CL
[5U1]  925:             mov   cl,bl
[KP4]  926:           ; Sekunden in DH, Hundertstel
[441]  927:           ; in DL
[3L2]  928:             mov   second,dh
[FH5]  929:           ; Alle drei Werte speichern, damit
[XM2]  93Ø:             mov   minute,cl
[Z75]  931:           ; sie später ausgegeben werden
[H12]  932:             mov   hour,ch
[3C1]  933:           ; können
[3PØ]  934:             ret
[UC1]  935:           ; Fertig.
[RK2]  936: make_time   endp
[F64]  937: ;-------------------------------------
[ØØØ]  938:
[1X3]  939: show_timdat proc  near
[D15]  94Ø:           ; Aufgabe: Zeigt die gefundenen
[Z45]  941:           ; Werte für Zeit und Datum an.
[E82]  942:             call  carriage
[E82]  943:             call  carriage
[F64]  944: ;-------------------------------------
[ZG3]  945: show_ftd    proc  near
[275]  946:           ; Aufgabe: dito, aber ohne Zeilen-
[T53]  947:           ; vorschub am Anfang
[NF2]  948:             msi   time_txt
[R75]  949:           ; Erstmal den Anfangs-Text raus.
[ND1]  95Ø:             call  txt
[222]  951:             mov   dx,2Ø3ah
[495]  952:           ; dann : und ein Leerzeichen raus
[R82]  953:             call  two_txt
[GV1]  954:             mov   al,day
[932]  955:           ; Tag laden...
[982]  956:             call  show_it
[LE2]  957:           ; ...und zeigen
[S92]  958:             mov   al,month
[PD2]  959:           ; Monat laden..
[8S2]  96Ø:             call  lead_punkt
[E95]  961:           ; ...und zeigen (mit Punkt davor)
[R22]  962:             mov   ax,year
[Z62]  963:           ; Jahr laden...
[PP1]  964:             mov   dl,1ØØ
[HW3]  965:           ; 'Hunderter' erzeugen...
[R61]  966:             div   dl
[WD5]  967:           ; durch 1ØØ teilen: Ergebnis in AL
[F91]  968:             push  ax
[2M3]  969:           ; Der 'Rest' steht in AH
[8S2]  97Ø:             call  lead_punkt
[CL4]  971:           ; 'Hunderter' mit Punkt raus
[971]  972:             pop   ax
```

```
[CX3]  973:           ; wieder Ergebnis laden
[AT1]  974:             mov   al,ah
[S93]  975:           ; und 'Rest' ausgeben
[982]  976:             call  show_it
[EH4]  977:           ; diesmal ohne Punkt davor...
[522]  978:             mov   dx,202ch
[2S4]  979:           ; jetzt Komma und Space raus
[R82]  980:             call  two_txt
[M22]  981:             mov   al,hour
[FV2]  982:           ; Stunden laden...
[982]  983:             call  show_it
[ME2]  984:           ; ... und zeigen.
[UK2]  985:             mov   al,minute
[BL2]  986:           ; dito Minuten...
[8S2]  987:             call  lead_punkt
[L24]  988:           ; ...aber mit Punkt davor.
[FH2]  989:             mov   al,second
[7U2]  990:           ; dito Sekunden...
[8S2]  991:             call  lead_punkt
[MN4]  992:           ; ...wieder mit Punkt davor.
[3PØ]  993:             ret
[F64]  994: ;-------------------------------------
[VP3]  995: lead_Punkt  proc  near
[3Ø5]  996:           ; UnterProgramm: Sichert eine
[WT4]  997:           ; Zahl (in Al), gibt dann Punkt
[YØ5]  998:           ; aus, gibt dann die gesicherte
[225]  999:           ; Zahl über das Unterprogramm
[974] 1ØØØ:           ; 'show_it' aus. Zahl wird
[695] 1Ø01:           ; durch push und pop gesichert.
[F91] 1Ø02:             push  ax
[LM1] 1Ø03:             mov   dl,"."
[672] 1Ø04:             call  one_txt
[971] 1Ø05:             pop   ax
[982] 1Ø06:             call  show_it
[3PØ] 1Ø07:             ret
[Y91] 1Ø08:           ; Fertig
[5S2] 1Ø09: lead_punkt  endp
[F64] 1Ø1Ø: ;-------------------------------------
[C85] 1Ø11:           ; UnterProgramm: Gibt eine Zahl
[4G3] 1Ø12: show_it     proc  near
[MA4] 1Ø13:           ; (zwei Ziffern) in AL aus.
[TLØ] 1Ø14:             aam
[AØ4] 1Ø15:           ; Zahl zu BCD Umwandlung.
[CU1] 1Ø16:             or    ax,3Ø3Øh
[EF4] 1Ø17:           ; ASCII-Zahl daraus machen.
[BS1] 1Ø18:             xchg  al,ah
[DK3] 1Ø19:           ; Ziffern vertauschen
[2W1] 1Ø2Ø:             mov   dx,ax
[WZ2] 1Ø21:           ; in DX schieben...
[R82] 1Ø22:             call  two_txt
[3M2] 1Ø23:           ; ...und ausgeben
[3PØ] 1Ø24:             ret
[Y91] 1Ø25:           ; Fertig
[EH2] 1Ø26: show_it     endp
[F64] 1Ø27: ;-------------------------------------
[9K2] 1Ø28: show_ftd    endp
[EM4] 1Ø29:           ; Erst müssen alle 'inneren'
[BY2] 1Ø3Ø: show_timdat endp
[CZ3] 1Ø31:           ; Procs beendet werden.
[F64] 1Ø32: ;-------------------------------------
[2S3] 1Ø33: show_usage  proc  near
[YF4] 1Ø34:           ; Aufgabe: Gibt die Hilfe aus
[ND1] 1Ø35:             call  txt
[E82] 1Ø36:             call  carriage
[E82] 1Ø37:             call  carriage
[YT2] 1Ø38:             msi   start_txt
[ØF5] 1Ø39:           ; Diese vielen Textausgaben sind
[HR4] 1Ø4Ø:           ; die Folge davon, daß *kein*
[K45] 1Ø41:           ; Wort im Text zweimal angegeben
[X85] 1Ø42:           ; ist. So ist die Textausgabe aber
[W64] 1Ø43:           ; sehr flexibel möglich.
[ND1] 1Ø44:             call  txt
[ØT1] 1Ø45:             mov   dx,"/["
[R82] 1Ø46:             call  two_txt
[S82] 1Ø47:             msi   opt_txt
[ND1] 1Ø48:             call  txt
[8R2] 1Ø49:             msi   usage_txt
[ND1] 1Ø5Ø:             call  txt
[SF5] 1Ø51:           ;Der Namen-Text wird nicht gleich
[445] 1Ø52:           ;am Anfang des Programms ausge-
[4G5] 1Ø53:           ;geben, weil sich durch die Datei-
[1K5] 1Ø54:           ;Bearbeitung im Programm dann ein
[MU5] 1Ø55:           ;'häßlicher' Zeitverzug bei der
[3T5] 1Ø56:           ;Text-Ausgabe einstellen könnte.
[E82] 1Ø57:             call  carriage
[E82] 1Ø58:             call  carriage
[S82] 1Ø59:             msi   opt_txt
[ND1] 1Ø6Ø:             call  txt
[9T2] 1Ø61:             msi   usage2_txt
[3PØ] 1Ø62:             ret
[CT2] 1Ø63: show usage  endp
[F64] 1Ø64: ;-------------------------------------
[ZØ2] 1Ø65: code        ends
[TX3] 1Ø66:           ; Code-Segment endet hier
[Z92] 1Ø67:             end   resident
[Ø93] 1Ø68:           ; Progamm endet hier
```

Listing 1. Mit »rtimer.com« können Sie auch ohne Echtzeituhr die Systemzeit setzen und vollautomatisch die Dauer von Arbeitssitzungen protokollieren

```
[695]     1: E9 CD Ø1 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ 63 3A 5C 72 74
[AG5]     2: 69 6D 65 ØØ ØB Ø1 44 Ø4 ØØ 9C 2E FF ØE 16 Ø1 75
[CH5]     3: ØD 2E C6 Ø6 18 Ø1 Ø1 2E C7 Ø6 16 Ø1 44 Ø4 9D 2E
[6C5]     4: FF 2E Ø3 Ø1 2E 8Ø 3E 18 Ø1 ØØ 74 23 5Ø 53 51 52
[7P5]     5: 1E ØE 1F 8B 16 14 Ø1 B4 3C 33 C9 CD 21 72 Ø6 8B
[7L5]     6: D8 B4 3E CD 21 C6 Ø6 18 Ø1 ØØ 1F 5A 59 5B 58 2E
[355]     7: FF 2E Ø7 Ø1 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ 26
[655]     8: ØØ ØØ ØØ ØØ ØØ ØØ ØØ 52 54 49 4D 45 52 2Ø ØØ 76
[P85]     9: 31 2E 3Ø 2Ø 2D 2Ø ØØ 28 63 29 2Ø 31 39 39 31 2Ø
[SF5]    1Ø: 44 4F 53 2Ø 49 6E 74 65 72 6E 61 74 69 6F 6E 61
[GK5]    11: 6C 2C 2Ø 41 75 74 6F 72 3A 2Ø 4A 6F 63 68 65 6E
[2G5]    12: 2Ø 42 6C 65 69 63 68 72 6F 74 68 ØD ØA 2D 2Ø 55
[3F5]    13: 6E 67 81 6C 74 69 67 65 72 2Ø 5Ø 61 72 61 6D 65
[9F5]    14: 74 65 72 2Ø 2D ØØ 5Ø 72 6F 67 72 61 6D 6D 2Ø 69
[NK5]    15: 6E 73 74 61 6C 6C 69 65 72 74 ØØ 4F 7Ø 74 69 6F
[MG5]    16: 6E 65 6E ØØ 5D 2Ø 5B 5B 5Ø 66 61 64 5D 44 61 74
[AE5]    17: 65 69 6E 61 6D 65 5D ØØ 3A 2Ø 53 3D 53 65 74 2Ø
[SC5]    18: 53 79 73 54 2F 44 2Ø 2Ø 52 3D 2F 53 26 49 6E 73
[ØF5]    19: 74 2E 2Ø 2Ø 46 3D 53 68 6F 77 2Ø 46 69 6C 65 54
[EH5]    2Ø: 2F 44 2Ø 2Ø 4E 3D 4E 6F 2Ø 54 65 78 74 2Ø 2Ø 6E
[5D5]    21: 6E 3D 44 65 6C 61 79 2Ø 31 2E 2E 31 3Ø ØØ 42 65
[XK5]    22: 69 2Ø 49 6E 73 74 61 6C 6C 61 74 69 6F 6E 2Ø 5Ø
[1C5]    23: 66 61 64 2Ø 6D 69 74 2Ø 61 6E 67 65 62 65 6E ØØ
[6C5]    24: 2Ø 28 44 61 74 65 69 2D 57 65 72 74 65 29 ØØ 46
[HB5]    25: 65 68 6C 65 72 2Ø 62 65 69 6D 2Ø ØØ 53 63 68 72
[4E5]    26: 65 69 62 2D ØØ 5A 75 67 72 69 66 66 2Ø 61 75 66
[CE5]    27: 3A 2Ø 27 ØØ 53 65 74 7A 65 6E 2Ø 76 6F 6E 2Ø ØØ
[PE5]    28: 53 79 73 74 65 6D 2D 44 61 74 75 6D 2Ø 75 6E 64
[GB5]    29: 2Ø 5A 65 69 74 ØØ 2Ø 67 65 73 65 74 7A 74 2Ø ØØ
[PL5]    3Ø: E8 EØ Ø1 8Ø 3E 69 Ø1 FF 75 Ø3 E9 Ø7 Ø1 8Ø 3E 6E
[ZM5]    31: Ø1 FF 74 ØA 8Ø 3E 68 Ø1 FF 75 22 E9 A9 ØØ E8 49
[UL5]    32: Ø1 72 3D 89 ØE 64 Ø1 89 16 66 Ø1 E8 CA Ø2 E8 9C
[4M5]    33: Ø2 E8 F6 Ø2 BE 7Ø Ø2 E8 8D Ø1 E9 16 Ø1 8Ø 3E 6A
[HK5]    34: Ø1 ØØ 74 Ø3 E9 CØ ØØ 8B 16 14 Ø1 B4 3C 33 C9 CD
[5K5]    35: 21 72 1Ø 8B D8 B4 3E CD 21 72 Ø8 E8 ØC Ø1 73 Ø6
[FN5]    36: E9 CC ØØ E9 BE ØØ 89 ØE 64 Ø1 89 16 66 Ø1 E8 87
[7D5]    37: Ø2 E8 59 Ø2 B8 Ø8 35 CD 21 89 1E Ø3 Ø1 8C Ø6 Ø5
[RK5]    38: Ø1 B8 Ø8 25 BA 19 Ø1 CD 21 B8 28 35 CD 21 89 1E
[LG5]    39: Ø7 Ø1 8C Ø6 Ø9 Ø1 B8 28 25 BA 34 Ø1 CD 21 8Ø 3E
[BN5]    4Ø: 6C Ø1 FF 74 Ø5 E8 E7 ØØ EB Ø9 8A 16 6F Ø1 B6 2Ø
[XP5]    41: E8 Ø9 Ø1 BE D6 Ø1 E8 ØE Ø1 E8 68 Ø2 E8 F8 ØØ BA
[9L5]    42: 17 ØØ B8 ØØ 31 CD 21 8Ø 3E 69 Ø1 FF 74 46 E8 99
[XL5]    43: ØØ 72 5C E8 F7 Ø1 B4 2B CD 21 3C ØØ 75 66 E8 17
[ØM5]    44: Ø2 B4 2D CD 21 3C ØØ 75 5B E8 A3 ØØ BE BØ Ø2 E8
[3N5]    45: D5 ØØ BE C6 Ø2 E8 CF ØØ 8Ø 3E 6C Ø1 FF 75 Ø3 E9
[ZR5]    46: 3B FF E8 1F Ø2 EB 4C C6 Ø6 6F Ø1 26 E8 8Ø ØØ BE
[2M5]    47: 4E Ø2 EB ØB C6 Ø6 6F Ø1 26 E8 73 ØØ BE 87 Ø1 E8
[8T5]    48: 62 Ø2 EB 2C E8 34 ØØ BE 8C Ø2 E8 9A ØØ EB Ø3 E8
[HH5]    49: 29 ØØ BE 95 Ø2 E8 8F ØØ 8B 36 14 Ø1 E8 88 ØØ BE
[DS5]    5Ø: A2 Ø2 EB ØC E8 14 ØØ BE A4 Ø2 E8 7A ØØ BE B7 Ø2
[CH5]    51: E8 74 ØØ E8 61 ØØ B8 Ø1 4C CD 21 C6 Ø6 6F Ø1 26
[GU5]    52: E8 2C ØØ BE 7F Ø2 E8 5E ØØ C3 B8 ØØ 3D 33 C9 8B
[9L5]    53: 16 14 Ø1 CD 21 72 17 8B D8 B8 ØØ 57 CD 21 72 ØE
[HF5]    54: 89 ØE 64 Ø1 89 16 66 Ø1 B4 3E CD 21 72 ØØ C3 8Ø
[YK5]    55: 3E 6D Ø1 ØØ 74 ØF E8 1E ØØ BE Ø7 ØØ 33 D2 E8 1B
[HL5]    56: ØØ 4E 75 FA C3 C6 Ø6 6D Ø1 FF BE 77 Ø1 E8 17 ØØ
[RS5]    57: BE 7F Ø1 E8 11 ØØ C3 BA ØA ØD 86 D6 B4 Ø2 CD 21
[PN5]    58: 86 D6 B4 Ø2 CD 21 C3 8Ø 3E 6F Ø1 26 74 Ø6 81 FE
[HP5]    59: B7 Ø2 75 ØE FC B4 Ø2 AC ØA CØ 74 Ø6 8A DØ CD 21
[4U5]    6Ø: EB F5 C3 BE 8Ø ØØ 8A 1C ØA DB 74 37 32 FF BE 82
[ZN5]    61: ØØ Ø3 DE FC AC 3B F3 77 12 3C 2Ø 74 Ø8 3C ØD 74
[GT5]    62: Ø4 3C ØA 7,5 EF C6 44 FF ØØ EB E9 BE 82 ØØ EB Ø3
[LR5]    63: E8 11 ØØ AC 3B F3 77 ØB 3C ØØ 74 F7 3C 2F 74 FØ
[ZM5]    64: E8 8B ØØ C3 8Ø 3E 6B Ø1 FF 74 16 AD 3C 39 77 Ø8
[CD5]    65: 3C 31 72 Ø4 E8 4A ØØ C3 8Ø FC ØØ 75 Ø4 E8 Ø7 ØØ
[ML5]    66: C3 C6 Ø6 69 Ø1 FF C3 24 DF 3C 53 75 ØB C6 Ø6 68
[AN5]    67: Ø1 FF C6 Ø6 6B Ø1 FF C3 3C 52 75 ØB C6 Ø6 6C Ø1
[DR5]    68: FF C6 Ø6 68 Ø1 FF C3 3C 46 75 Ø6 C6 Ø6 6E Ø1 FF
[HM5]    69: C3 3C 4E 75 Ø6 C6 Ø6 6F Ø1 2Ø C3 C6 Ø6 69 Ø1 FF
[TL5]    7Ø: C3 8Ø FC ØØ 74 12 8Ø FC 3Ø 75 1D 3C 31 75 19 AC
[ØL5]    71: 3C ØØ 74 Ø2 EB 12 BØ 3A 34 3Ø 32 E4 B9 44 Ø4 F7
[8L5]    72: E1 A3 2C Ø1 A3 16 Ø1 C3 C6 Ø6 69 Ø1 FF C3 AC 4E
[HL5]    73: 3C 3A 74 13 8Ø 3E 6C Ø1 FF 74 Ø7 8Ø 3E 68 Ø1 FF
[7H5]    74: 74 Ø5 C6 Ø6 6A Ø1 FF 4E 89 36 14 Ø1 C3 A1 66 Ø1
[HP5]    75: 8B D8 8B DØ B1 Ø9 D3 EB 81 C3 BC Ø7 25 EØ Ø1 B1
[2M5]    76: Ø5 D3 E8 81 E2 1F ØØ 8A FØ 8B CB 88 16 73 Ø1 88
[4L5]    77: 36 74 Ø1 89 ØE 75 Ø1 C3 A1 64 Ø1 8B D8 8B DØ B1
[5K5]    78: ØB D3 E8 81 E3 EØ Ø7 B1 Ø5 D3 EB 81 E2 1F ØØ Ø3
[FN5]    79: D2 86 D6 8A E8 8A CB 88 36 7Ø Ø1 88 ØE 71 Ø1 88
[KX5]    8Ø: 2E 72 Ø1 C3 E8 9Ø FE E8 8D FE BE B7 Ø2 E8 97 FE
[1L5]    81: BA 3A 2Ø E8 86 FE AØ 73 Ø1 E8 3B ØØ AØ 74 Ø1 E8
[HM5]    82: 2A ØØ A1 75 Ø1 B2 64 F6 F2 5Ø E8 1F ØØ 58 8A C4
[3M5]    83: E8 24 ØØ BA 2C 2Ø E8 63 FE AØ 72 Ø1 E8 18 ØØ AØ
[1E5]    84: 71 Ø1 E8 Ø7 ØØ AØ 7Ø Ø1 E8 Ø1 ØØ C3 5Ø B2 2E E8
[YK5]    85: 5Ø FE 58 E8 Ø1 ØØ C3 D4 ØA ØD 3Ø 3Ø 86 C4 8B DØ
[FZ5]    86: E8 39 FE C3 E8 4Ø FE E8 2D FE E8 2A FE BE 77 Ø1
[C26]    87: E8 34 FE BA 5B 2F E8 23 FE BE EB Ø1 E8 28 FE BE
[SZ5]    88: F4 Ø1 E8 22 FE E8 ØF FE E8 ØC FE BE EB Ø1 E8 16
[PP1]    89: FE BE Ø8 Ø2 C3
```

Listing 2. Eine Fundgrube für Assembler-Programmierer: »rtimer.asm«



MASM ab Version 4.0

Umlaute für den Laser

Welcher Besitzer eines Laserdruckers kennt nicht das Problem, daß zahlreiche Programme beim Ausdruck keine Codewandlung durchführen und so die Umlaute nicht dargestellt werden? Noch ärger wird die Situation, wenn Matrix- und Laserdrucker am gleichen Computer, gegebenenfalls sogar im Netzwerk verwendet werden. Unser Listing »hp.com« bewährt sich in all diesen Fällen.

Die Vielfalt bei Druckern bezüglich Ansteuerung, Zeichensätzen und nicht realisierten Normierungsversuchen verursachen Probleme, seitdem sie an den PC angeschlossen werden. Von Typenrad- über Matrix- bis hin zu postscriptfähigen Laserdruckern erstreckt sich die Palette. Für fast jede technische Unzulänglichkeit oder Besonderheit gibt es ein spezielles Hilfsprogramm, das den Fehler beheben oder die Besonderheit realisieren soll. Leider verursachen diese Programme in den individuell gestalteten Betriebssystemumgebungen häufig erneute Probleme. Spätestens aber, wenn zwei Drucker gleichzeitig, zum Beispiel ein Matrix- und ein Laserdrucker, an einen PC angeschlossen werden, ist der Einsatz eines solchen Hilfsprogramms fragwürdig, da es fast ausschließlich für einen Zweck konzipiert wurde. Stehen Drucker über ein Netzwerk zur Verfügung, versagen die meisten Softwarewerkzeuge völlig. Dieser Artikel gibt eine Übersicht über die am häufigsten gemachten Fehler und versucht anhand eines praxisorientierten Programmierbeispiels einen sicheren Weg zur Programmierung solcher Utility-Programme aufzuzeigen. Druckerprogramme sollen eine Anpassung zwischen einem verwendeten Drucker und dem jeweils eingesetzten PC auch dann noch gewährleisten, wenn zusätzliche Programme geladen werden. Hierzu ist es also erforderlich, solche Druckertools permanent im Speicher zu halten, das heißt, sie sollten speicherresident gemacht werden. Diese TSR-Programme haben nicht nur die wünschenswerte Eigenschaft, daß sie permanent im Computer ihre Arbeit verrichten, sie haben auch die unerwünschte Eigenschaft, daß sie den innerhalb eines PC zu knapp dimensionierten Speicher permanent um den selbst benötigten Platz reduzieren. Also gilt es, mit den Speicherressourcen sehr sparsam umzugehen. Aus diesem Grunde verwendet man für die speicherresidente Programmierung in der Regel Assembler. Bei der Assembler-Programmierung kann der Programmierer auf jeden einzelnen Prozessorbefehl direkten Einfluß nehmen. Da speicherresidente Programme aber permanent im Arbeitsspeicher liegen, ist hier die Fehleranfälligkeit besonders hoch. Wie man dennoch solide residente Assembler-Programme schreibt, zeigt das diesem Artikel zugrundeliegende Programm »hp.asm« (Listing 1). Oftmals benötigen aber selbst Assembler-Programme in Verbindung mit größeren Anwendungsprogrammen mehr Speicherplatz als im Computer zur Verfügung steht. Eine zweite Methode, Speicherplatz zu sparen, besteht darin, die speicherresidenten Programme nur dann zu laden, wenn sie auch wirklich benötigt werden. Leider lassen das die meisten Programme nur über einen Warmstart des PCs zu. Dabei kann

man sich mit einem ganz einfachen Trick behelfen: TSR-Programme werden in der Regel nicht explizit durch den Benutzer, sondern über Unterbrechungen (Interrupts) aufgerufen. Ein Programm hält seine Arbeit an und sendet eine Unterbrechung an das Betriebssystem, zum Beispiel, um mit dem Drucker zu kommunizieren. Ist ein speicherresidentes Programm für diese Unterbrechung zuständig, so kann es dem Betriebssystem seine Speicheradresse mitteilen, an der die Prozessorarbeit fortgesetzt werden soll. Sämtliche Adressen für solche Unterbrechungen werden in einer Tabelle von MS-DOS verwaltet, und jede einzelne Adresse kann über die DOS-Funktion 35hex (»Get Vector«) abgefragt werden. Speichert man nun innerhalb einer bestimmten Adresse des speicherresidenten Programms eine beliebige Kennung, in unserem Beispiel sind es die zwei Buchstaben »hp«, so kann das Betriebssystem anhand dieser Kennung ein Programm identifizieren. Das TSR-Codewandlungsprogramm »hp.com« (Listing 2) benutzt diese Kennung beispielsweise am Anfang des Aufrufs, um festzustellen, ob es schon resident im Arbeitsspeicher installiert ist. Damit wird erstens der Fehler vermieden, daß sich ein und dasselbe Programm im Arbeitsspeicher mehrfach installiert, und zweitens hat das Programm die Fähigkeit, beim erneuten Aufruf zu entscheiden, ob es sich dann entsprechend zu deinstallieren hat.

Selbstidentifikation durch Kennung

Dabei benötigt man lediglich drei Assembler-Zeilen zum Lesen der Adresse eines DOS-Interrupts und drei weitere Zeilen zum Vergleichen der Kennung. Als Referenz kann hierzu das Label »inst« im Beispielprogramm angegeben werden.

Nachdem nun festgestellt wurde, daß das Programm noch nicht im Arbeitsspeicher installiert ist, gilt es zunächst, die Voraussetzungen für eine korrekte Deinstallation des Programms zu schaffen. Hierbei können fatale Fehler gemacht werden. Bei der Freigabe des Arbeitsspeichers muß darauf geachtet werden, daß die Adresse der DOS-Unterbrechung einen korrekten Wert besitzt. Also sichert man die alte Adresse der ursprünglichen DOS-Unterbrechung innerhalb des Speicherplatzes des residenten Programms. Würde man später den Speicher freigeben, ohne die Unterbrechungsadresse zu korrigieren, so würde der Prozessor mit einem vom Zufall abhängigen Befehl fortfahren. Mit größter Wahrscheinlichkeit würde dies mit einem Systemabsturz enden. Ein häufig gemachter Fehler bei der Installation von speicherresidenten Programmen ist die fehlende Sicherung der Umgebungsadresse. DOS reserviert für jedes Programm eine eigene DOS-Umgebung von 128 Byte. Innerhalb dieser Umgebung gibt der Benutzer eventuelle Parameter beim Aufruf des Programms ein. Bei der Freigabe des benutzten Arbeitsspeichers wird die Freigabe der DOS-Umgebung von residenten Programmen häufig vergessen. Dies führt zwar nicht zu Problemen beim weiteren Arbeiten am PC, lediglich der Arbeitsspeicher wird bei jedem erneuten Aufruf des residenten Programms um weitere 128 Byte verringert. Ist nun die Adresse der alten Unterbrechung sowie die Umgebungsadresse (Environment-Adresse) gesichert, sollte das Programm noch eine Installationsmeldung ausgeben und sich anschließend resident machen: Das Programm teilt MS-DOS nun die neue Unterbrechungsadresse mit und macht sich mit der DOS-Unterbrechung 27hex resident. Dabei wird nur der Speicherbereich resident gemacht, der vor dem Label »inst« liegt, denn die Installationsroutine wird nur bei der Installation des Programms benötigt und gehört damit zum nichtresidenten Teil des Beipielprogramms.

Als neue Adresse der DOS-Unterbrechung wird das Label »int17« angegeben. Aus dem Label läßt sich ableiten, daß Sie den DOS-Interrupt 17hex verändern. Diese Unterbrechung wird immer bei der Ausgabe von Zeichen auf Druckern aufgerufen. Zunächst überprüft das Programm, ob es sich beim Aufruf lediglich um eine Funktion handelt, die nur den Status des Druckers prüfen soll.

Der erste Schlüssel der Codewandlung: Interrupt 17hex

Das Register »AH« des Prozessors enthält in diesem Fall den Wert Null. Manche Programme prüfen vor jeder Ausgabe auf den Drucker, ob diese Peripherieeinheit betriebsbereit ist. Auch wenn diese Programme durch unser Hilfsprogramm keine Schwierigkeiten bekommen, so scheitern sie fast alle in PC-Netzwerken mit Drucker-Server. Bei manchen Programmen hilft der lokale Anschluß eines beliebigen, betriebsbereiten Druckers, während die Ausgabe dann aber dennoch auf dem qualitativ besseren Laserdrucker im Netz erfolgt. »hp« läßt die Kommunikation über den Druckerstatus zu, indem es die Anfrage an die Adresse der ursprünglichen DOS-Unterbrechung weiterleitet. Eine weitere Funktion wird nicht ausgeführt.

Wird nun eine Funktion von residenten Programmen verlangt, so werden zunächst sämtliche Prozessorregister, die während der Ausführung einer Funktion verändert werden können, gesichert. Damit erhält das Programm, das den Interrupt ausgelöst hat, nur die Register verändert zurück, die zur Kommunikation notwendig sind. Verwendet der angeschlossene Drucker nun einen anderen Zeichensatz als den des IBM-PC, so können hierfür Übersetzungstabellen angelegt werden.

Wandeltabellen erledigen den Rest

Solche Tabellen sind in »hp« für die deutschen Umlaute bei Epson-kompatiblen Druckern und Laserdruckern, die häufig den romanischen Zeichensatz verwenden, angelegt. Bei Laserdruckern verwenden insbesondere die oft eingesetzten Softfonts wie »Times Roman« oder »Helvetica« keinen IBM-Zeichensatz; Umlaute, »ß« etc. sind an anderer Stelle als im IBM-Zeichensatz zu finden.

Die entsprechenden Wandlungstabellen können ganz einfach angelegt werden: Zuerst gibt man die Zeichen an, die konvertiert werden sollen. Anschließend gibt man in einem zweiten Vektor an, in welche Zeichen konvertiert werden soll. Mit dem Assembler-Befehl »REPNZ SCASB« sucht der Prozessor »CX«-mal das Zeichen in »AH« ab der Adresse in »DI«. Ist das Zeichen in »AH« zu konvertieren, so wird entsprechend mit dem Zeichen getauscht, dessen Adresse sich aus dem Register »DI« ableiten läßt. Bei diesem Verfahren lassen sich sowohl die zu konvertierenden Zeichen als auch die korrespondierenden Tabellen einfach ändern. Beliebige Ergänzungen sind hier möglich. »hp« kennt den Status »e« »Epson«, »r« »Roman-8« und »t« (transparent, keine Wandlung).

Dynamischer Programmeinsatz durch Statusänderung

Der Status des Programms läßt sich durch den Aufruf über das Betriebssystem mit entsprechendem Parameterzusatz (»/e«, »/r«, »/t«) jederzeit ändern und umschalten. Zunächst stellt das Programm über das oben geschilderte Verfahren fest, ob es überhaupt installiert ist. Sofern dies der Fall ist, befindet sich die Kommandozeile im gleichen Codesegment wie das Programm selber, und es läßt sich abfragen, ob ein Parameter eingegeben wurde. Durch diese Art der Umschaltung entfällt das leidige Problem, daß bei Ausdrucken von Grafiken residente Programme irgendeine Art von Konvertierung vornehmen und die Grafiken dadurch verfälscht werden. Außerdem können durch das Umschalten,

gleichzeitig mehrere verschiedene Drucker eingesetzt werden, wobei jeder Drucker dabei seine eigene Konvertierungstabelle besitzen kann.
Nach erfolgreicher Konvertierung des Zeichens in AH ruft das Programm anschließend noch die ursprünglich gespeicherte Adresse des DOS-Interrupts auf. Diese Vorgehensweise hat den Zweck, daß auch mehrere Druckerhilfsprogramme koexistieren können. Beipielsweise kann »hp« durch einen residenten Drucker-Spooler unterstützt werden. Hierbei gilt insbesondere zu berücksichtigen, daß auch Netzwerkdrucker über residente Druckertools angeschlossen werden. Und wer möchte sein Hilfsprogramm nur ohne Netzwerk benutzen können? Die hier beschriebene Vorgehensweise erfreut sich schon mehrjähriger Praxis in einem Novell-Netzwerk.
Zum Schluß soll nochmals darauf hingewiesen werden, daß beim Deinstallieren des Programms die DOS-Funktion 49hex zweimal zur Freigabe von Speicher aufgerufen werden muß. Einmal zur Freigabe des vom Programm belegten Speichers und das andere Mal zur Freigabe des Environments, wobei sich der verwendete Speicher bei »hp« auf lediglich 496 Byte beschränkt. (Michael Lontke/bj)

```
[LZ2]    1: ; Programm: hp.asm
[B27]    2: ; Funktion: Residenter Druckertreiber zur
[NZ3]    3: ;  Zeichenkonvertierung
[D73]    4: ; Sprache: Tasm ab 1.Ø
[KY3]    5: ; Autor: Michael Lontke
[ZN4]    6: (c)1991 DMV Widuch GmbH & Co. KG
[P3Ø]    7: ;
[ØØØ]    8:
[5U4]    9: ;-------------------------------------------
[UG2]   1Ø: ;       CODESEGMENT
[5U4]   11: ;-------------------------------------------
[ØØØ]   12:
[KB3]   13: cseg     segment 'code'
[7V4]   14:          assume cs:cseg,ds:cseg,es:cseg
[ØØØ]   15:
[A61]   16:          org 1ØØh
[ØØØ]   17:
[UV2]   18: main:    JMP      inst
[ØØØ]   19:
[7Y6]   2Ø: ;====== DATA AT Codesegment ===============
[ØØØ]   21:
[8B2]   22: org_vec dd        ?
[452]   23: envir   dw        ?
[PV2]   24: kennung db        'HP'
[NL2]   25: ersetz  db        'r'
[ØØØ]   26:
[5U4]   27: ;-------------------------------------------
[SD5]   28: ;     Neuer Interrupt 17H (Drucker)
[5U4]   29: ;-------------------------------------------
[ØØØ]   3Ø:
[RU2]   31: int17    proc     far
[EMØ]   32:          CLI
[ØU2]   33:                   ; Interrupts aus
[ZV1]   34:          CMP      AH,ØØH
[9M2]   35:                   ; Printerfkt. Ø ?
[2X1]   36:          JNE      print
[ØØØ]   37:
[PF1]   38:          PUSH     ES
[PE1]   39:          PUSH     DI
[1G1]   4Ø:          PUSH     CX
[KF1]   41:          PUSH     CS
[GD1]   42:          POP      ES
[ØØØ]   43:
[N33]   44:          MOV      CL,CS:ersetz
[DU1]   45:          CMP      CL,'e'
[AM3]   46:                   ; Epson-Zeichensatz ?
[MW1]   47:          JNE      roman
[ØØØ]   48:
[ULØ]   49:          CLD
[ST3]   5Ø:          MOV      DI,offset ibm_pc
[162]   51:          MOV      CX,ØØØ7H
[ZZ1]   52:          REPNZ    SCASB
[TZ1]   53:          JNZ      nKonv
[7F6]   54:       MOV AL,ES:[offset ai-offset ib-1] [DI]
[EY1]   55:          JMP      nKonv
[ØØØ]   56:
[R23]   57: roman:   CMP      CL,'r'
[3W3]   58:                   ; Roman 8-Zeichensatz ?
[5X1]   59:          JNE      nKonv
[ØØØ]   6Ø:
[ULØ]   61:          CLD
[YZ2]   62:          MOV      DI,offset ib
[162]   63:          MOV      CX,ØØØ7H
[ZZ1]   64:          REPNZ    SCASB
[TZ1]   65:          JNZ      nKonv
[UG6]   66:       MOV AL,ES:[offset r8-offset ib-1] [DI]
```

```
[ØØØ]   67:
[WN2]   68: nKonv:   POP      CX
[GC1]   69:          POP      DI
[GD1]   7Ø:          POP      ES
[ØØØ]   71:
[TX6]   72: ;------ INT17H-Original-Interrupt aufrufen
[ØØØ]   73:
[4Y1]   74: print:   STI
[FV3]   75:                   ; Interrupts wieder an
[ZXØ]   76:          PUSHF
[UX3]   77:                   ; Interrupt simulieren
[U53]   78:          CALL     cs:[org_vec]
[5H4]   79:                   ; Originalroutine aufrufen
[EMØ]   8Ø:          CLI
[GT2]   81:                   ; interrupt aus
[ØØØ]   82:
[F42]   83: ;------ zurück
[ØØØ]   84:
[6PØ]   85:          STI
[FV3]   86:                   ; Interrupts wieder an
[GRØ]   87:          IRET
[ØØØ]   88:
[C25]   89: ;       Ä    Ö    Ü    ä    ö    ü    ß
[Ø45]   9Ø: ib DB 8Eh, 99h, 9Ah, 84h, 94h, 81h,ØE1h
[S75]   91: ai DB 5Bh, 5Ch, 5Dh, 7Bh, 7Ch, 7Dh, 7Eh
[FR5]   92: r8 DB ØD8h,ØDAh,ØDBh,ØCCh,ØCEh,ØCFh,ØDEh
[ZT1]   93: int17    endp
[ØØØ]   94:
[ØØØ]   95:
[5U4]   96: ;-------------------------------------------
[3F4]   97: ;       Installationsroutine
[5U4]   98: ;-------------------------------------------
[ØØØ]   99:
[MM2]  1ØØ: inst     proc near
[ØØØ]  1Ø1:
[CY1]  1Ø2:          MOV      AL,17h
[ZR2]  1Ø3:                   ; Vector Nummer
[YX1]  1Ø4:          MOV      AH,35h
[614]  1Ø5:                   ; DOS-Funktion get Vector
[3K1]  1Ø6:          INT      21h
[ØØØ]  1Ø7:
[PP4]  1Ø8: ;------ bereits installiert ?
[ØØØ]  1Ø9:
[PA5]  11Ø:          MOV      AX,WORD PTR ES:[kennung]
[SA5]  111:          MOV      DX,WORD PTR CS:[kennung]
[VT1]  112:          CMP      AX,DX
[2E2]  113:          JE       installed
[ØØØ]  114:
[LX4]  115: ;------ neuen Interrupt setzen
[ØØØ]  116:
[ED5]  117:          MOV      WORD PTR CS:[org_vec],BX
[1A4]  118:                   ; alten Interrupt sichern
[ZS5]  119:          MOV      WORD PTR CS:[org_vec+2],ES
[ØØØ]  12Ø:
[CY1]  121:          MOV      AL,17h
[TS2]  122:                   ; Vector-Nummer
[EK3]  123:          MOV      DX,offset int17
[YL4]  124:                   ; Adresse des neuen Vectors
[XX1]  125:          MOV      AH,25h
[K14]  126:                   ; DOS-Funktion set Vector
[3K1]  127:          INT      21h
[ØØØ]  128:
[E75]  129: ;------ Enviroment Adresse retten
[ØØØ]  13Ø:
[Ø43]  131:          MOV      AX,CS:[2Ch]
[G44]  132:                   ; Technical Reference DOS
[U33]  133:          MOV      CS:envir,AX
[ØØØ]  134:
[BG4]  135: ;------ Meldung installiert
[ØØØ]  136:
[DB3]  137:          MOV      DX,offset msg1
[BR1]  138:          MOV      AH,Ø9
[3K1]  139:          INT      21h
[ØØØ]  14Ø:
[5K5]  141: ;------ Terminate and stay resident
[ØØØ]  142:
[TE3]  143:          MOV      DX,offset Inst
[6PØ]  144:          STI
[NK1]  145:          INT      27h
[ØØØ]  146:
[DT1]  147: installed:
[ØØØ]  148:
[R86]  149: ;------ Parameter in der Kommandozeile ?
[ØØØ]  15Ø:
[P74]  151:          cmp      byte ptr CS:[8Øh],Ø
[EX1]  152:          je       deinst
[RG5]  153:          ; keine Parameter deinstallieren
[ØØØ]  154:
[3N5]  155: ;------ Länge des Parameterstrings
[ØØØ]  156:
[PV1]  157:          XOR      CX,CX
[VU1]  158:          XOR      SI,SI
[Ø23]  159:          MOV      CL,CS:[8Øh]
[ØØØ]  16Ø:
[KP3]  161: ;------ Delimiter suchen
[ØØØ]  162:
[894]  163: lpØ:     MOV      AH,CS:[81h+SI]
[HS1]  164:          CMP      AH,'/'
[FC1]  165:          JE       lp1
```

```
[9B1]  166:          INC     SI
[TM1]  167:          LOOP    lpØ
[ØØØ]  168:
[7W6]  169: ;------ das folgende Zeichen ist Ersetzung
[ØØØ]  17Ø:
[F92]  171: lp1:     INC     SI
[6B3]  172:          MOV     AH,CS:[81h+SI]
[S33]  173:          MOV     ES:ersetz,AH
[ØØØ]  174:
[4R1]  175:          JMP     ende
[ØØØ]  176:
[TP4]  177: ;------ Vector zurücksetzen
[ØØØ]  178:
[AP6]  179: deinst:  MOV     DX,WORD PTR ES:[org_vec]
[UM5]  18Ø:          MOV     AX,WORD PTR ES:[org_vec+2]
[1V1]  181:          MOV     DS,AX
[CY1]  182:          MOV     AL,17h
[HG3]  183:                  ; Printer-Interrupt
[XX1]  184:          MOV     AH,25h
[4P4]  185:                  ; DOS-Funktion Set Interrupt
[3K1]  186:          INT     21h
[ØØØ]  187:
[D93]  188: ;------ free codeblock
[ØØØ]  189:
[7Y1]  19Ø:          MOV     AH,49h
[3K1]  191:          INT     21h
[ØØØ]  192:
[ØR3]  193: ;------ free Enviroment
[ØØØ]  194:
[W13]  195:          MOV     ES,ES:envir
[7Y1]  196:          MOV     AH,49h
[3K1]  197:          INT     21h
[ØØØ]  198:
[KF1]  199:          PUSH    CS
[ED1]  2ØØ:          POP     DS
[742]  2Ø1:          JNC     res_ok
[ØØØ]  2Ø2:
[ER3]  2Ø3: ;------ Fehlermeldungen
[ØØØ]  2Ø4:
[XF1]  2Ø5:          PUSH    AX
[1Ø4]  2Ø6:          MOV     DX,offset err_msg
[BR1]  2Ø7:          MOV     AH,Ø9
[3K1]  2Ø8:          INT     21h
[RD1]  2Ø9:          POP     AX
[5P1]  21Ø:          CMP     AL,Ø9
[XH1]  211:          JE      err2
[ØØØ]  212:
[185]  213: err1:    MOV     DX,offset err_msg1
[BR1]  214:          MOV     AH,Ø9
[3K1]  215:          INT     21h
[7T1]  216:          JMP     term
[ØØØ]  217:
[585]  218: err2:    MOV     DX,offset err_msg2
[BR1]  219:          MOV     AH,Ø9
[3K1]  22Ø:          INT     21h
[ØØØ]  221:
[3C3]  222: term:    MOV     AL,Øffh
[4R1]  223:          JMP     ende
[ØØØ]  224:
[ØØØ]  225:
[X23]  226: ;------ Programmende
[ØØØ]  227:
[RW2]  228: res_ok:  PUSH    CS
[ED1]  229:          POP     DS
[GB3]  23Ø:          MOV     DX,offset msg2
[BR1]  231:          MOV     AH,Ø9
[3K1]  232:          INT     21h
[MT1]  233:          XOR     AL,AL
[ØØØ]  234:
[XTØ]  235: ende:
[6PØ]  236:          STI
[VY1]  237:          MOV     AH,4Ch
[3K1]  238:          INT     21h
[ØØØ]  239:
[GT1]  24Ø: inst     endp
[ØØØ]  241:
[1G2]  242: ;------ Meldungen
[ØØØ]  243:
[896]  244: msg1 db 13,1Ø,'(C) 1991 DOS International'
[D85]  245:      db 13,1Ø,'    Autor: Michael Lontke'
[EY4]  246:      db 13,1Ø,'HP-Roman installiert -'
[R94]  247:      db 'Druckersteuerzeichen'
[8M4]  248:      db ' werden übersetzt.',13,1Ø,
[PU3]  249:      db '/e für Epson-, /r für'
[RU3]  25Ø:      db 'Roman-Zeichensatz, '
[MA5]  251:      db '/t für keine Konvertierung.',
[DZ4]  252:      db 13,1Ø,'HP ohne Parameter zum'
[2A4]  253:      db 'Deinstallieren.',1Ø,13,'$'
[E46]  254: msg2 db 13,1Ø,'HP-Roman deinstalliert.'
[8E1]  255:      db ,13,1Ø,'$'
[7N5]  256: err_msg db 13,1Ø,'HP-Roman - FEHLER:'
[PU3]  257:         db Speicher kann nicht'
[T35]  258:         db ' freigemacht werden.',13,1Ø,'$'
[616]  259: err_msg1  db  13,1Ø,'HP-Roman - FEHLER:'
[HM4]  26Ø:           db  ' Memory Area zerstört:'
[AD5]  261:           db  ' Computer muß neu gebootet'
[9Ø3]  262:           db  ' werden!',13,1Ø,'$'
[716]  263: err_msg2  db  13,1Ø,'HP-Roman - FEHLER:'
[V95]  264:           db  ' Falsche Segmentzuweisung.',
[WD1]  265:           db  13,1Ø,'$'
```

```
[ØØØ]  266:
[HR1]  267: cseg     ends
[LN1]  268:          end     main
```

Listing 1. Das Assembler-Programm »hp.asm« zeigt die Technik und den Einsatz von frei definierbaren Codewandlungstabellen

```
[LC5]    1: EB 69 9Ø ØØ ØØ ØØ ØØ ØØ ØØ 48 5Ø 72 FA 8Ø FC ØØ
[CF5]    2: 75 3A Ø6 57 51 ØE Ø7 2E 8A ØE ØB Ø1 8Ø F9 65 75
[SH5]    3: 13 FC BF 56 Ø1 B9 Ø7 ØØ F2 AE 75 1D 26 8A 85 Ø6
[ED5]    4: ØØ EB 16 9Ø 8Ø F9 72 75 1Ø FC BF 56 Ø1 B9 Ø7 ØØ
[WP5]    5: F2 AE 75 Ø5 26 8A 85 ØD ØØ 59 5F Ø7 FB 9C 2E FF
[ST5]    6: 1E Ø3 Ø1 FA FB CF 8E 99 9A 84 94 81 E1 5B 5C 5D
[H26]    7: 7B 7C 7D 7E D8 DA DB CC CE CF DE BØ 17 B4 35 CD
[VE5]    8: 21 26 A1 Ø9 Ø1 2E 8B 16 Ø9 Ø1 3B C2 74 28 2E 89
[BL5]    9: 1E Ø3 Ø1 2E 8C Ø6 Ø5 Ø1 BØ 17 BA ØC Ø1 B4 25 CD
[LH5]   1Ø: 21 2E A1 2C ØØ 2E A3 Ø7 Ø1 BA 27 Ø2 B4 Ø9 CD 21
[GL5]   11: BA 6B Ø1 FB CD 27 2E 8Ø 3E 8Ø ØØ ØØ 74 24 33 C9
[RH5]   12: 33 F6 2E 8A ØE 8Ø ØØ 2E 8A A4 81 ØØ 8Ø FC 2F 74
[KL5]   13: Ø3 46 E2 F3 46 2E 8A A4 81 ØØ 26 88 26 ØB Ø1 EB
[WE5]   14: 51 9Ø 26 8B 16 Ø3 Ø1 26 A1 Ø5 Ø1 8E D8 BØ 17 B4
[SG5]   15: 25 CD 21 B4 49 CD 21 26 8E Ø6 Ø7 Ø1 B4 49 CD 21
[6G5]   16: ØE 1F 73 23 5Ø BA 2B Ø3 B4 Ø9 CD 21 58 3C Ø9 74
[BR5]   17: ØA BA 6A Ø3 B4 Ø9 CD 21 EB Ø8 9Ø BA B9 Ø3 B4 Ø9
[CV5]   18: CD 21 BØ FF EB ØC 9Ø ØE 1F BA ØF Ø3 B4 Ø9 CD 21
[HK5]   19: 32 CØ FB B4 4C CD 21 ØD ØA 28 43 29 2Ø 31 39 39
[EK5]   2Ø: 31 2Ø 44 4F 53 2Ø 49 6E 74 65 72 6E 61 74 69 6F
[XD5]   21: 6E 61 6C ØD ØA 2Ø 2Ø 2Ø 2Ø 41 75 74 6F 72 3A 2Ø
[8K5]   22: 4D 69 63 68 61 65 6C 2Ø 4C 6F 6E 74 6B 65 ØD ØA
[8K5]   23: 48 5Ø 2D 52 6F 6D 61 6E 2Ø 69 6E 73 74 61 6C 6C
[GD5]   24: 69 65 72 74 2Ø 2D 2Ø 44 72 75 63 6B 65 72 73 74
[GD5]   25: 65 75 65 72 7A 65 69 63 68 65 6E 2Ø 77 65 72 64
[8D5]   26: 65 6E 2Ø 81 62 65 72 73 65 74 7A 74 2E ØD ØA ØØ
[BH5]   27: 2F 65 2Ø 66 81 72 2Ø 45 7Ø 73 6F 6E 2D 2C 2Ø 2F
[5H5]   28: 72 2Ø 66 81 72 2Ø 52 6F 6D 61 6E 2D 5A 65 69 63
[3D5]   29: 68 65 6E 73 61 74 7A 2C 2Ø 2F 74 2Ø 66 81 72 2Ø
[6L5]   3Ø: 6B 65 69 6E 65 2Ø 4B 6F 6E 76 65 72 74 69 65 72
[9F5]   31: 75 6E 67 2E ØØ ØD ØA 48 5Ø 2Ø 6F 68 6E 65 2Ø 5Ø
[CF5]   32: 61 72 61 6D 65 74 65 72 2Ø 7A 75 6D 2Ø 44 65 69
[8M5]   33: 6E 73 74 61 6C 6C 69 65 72 65 6E 2E ØA ØD 24 ØD
[ZH5]   34: ØA 48 5Ø 2D 52 6F 6D 61 6E 2Ø 64 65 69 6E 73 74
[7M5]   35: 61 6C 6C 69 65 72 74 2E ØD ØA 24 ØD ØA 48 5Ø 2D
[ED5]   36: 52 6F 6D 61 6E 2Ø 2D 2Ø 46 45 48 4C 45 52 3A 2Ø
[FG5]   37: 53 7Ø 65 69 63 68 65 72 2Ø 6B 61 6E 6E 2Ø 6E 69
[HB5]   38: 63 68 74 2Ø 66 72 65 69 67 65 6D 61 63 68 74 2Ø
[SF5]   39: 77 65 72 64 65 6E 2E ØD ØA 24 ØD ØA 48 5Ø 2D 52
[WH5]   4Ø: 6F 6D 61 6E 2Ø 2D 2Ø 46 45 48 4C 45 52 3A 2Ø 4D
[VD5]   41: 65 6D 6F 72 79 2Ø 41 72 65 61 2Ø 7A 65 72 73 74
[4F5]   42: 94 72 74 3A 2Ø 43 6F 6D 7Ø 75 74 65 72 2Ø 6D 75
[ØE5]   43: E1 2Ø 6E 65 75 2Ø 67 65 62 6F 6F 74 65 74 2Ø 77
[SG5]   44: 65 72 64 65 6E 21 ØD ØA 24 ØD ØA 48 5Ø 2D 52 6F
[SD5]   45: 6D 61 6E 2Ø 2D 2Ø 46 45 48 4C 45 52 3A 2Ø 46 61
[ØK5]   46: 6C 73 63 68 65 2Ø 53 65 67 6D 65 6E 74 7A 75 77
[HN3]   47: 65 69 73 75 6E 67 2E ØØ ØD ØA 24
```

Listing 2. Das speicherresidente Codewandlungsprogramm »hp.com« befähigt zur Ausgabe von Umlauten auch bei HP-Laser-Softfonts

Wir kämpfen für

WENIGER MÜLL!

Ich möchte mehr über Greenpeace wissen!
Für Ihre Kosten habe ich 3,60 DM in Briefmarken beigelegt:

Vorname, Name

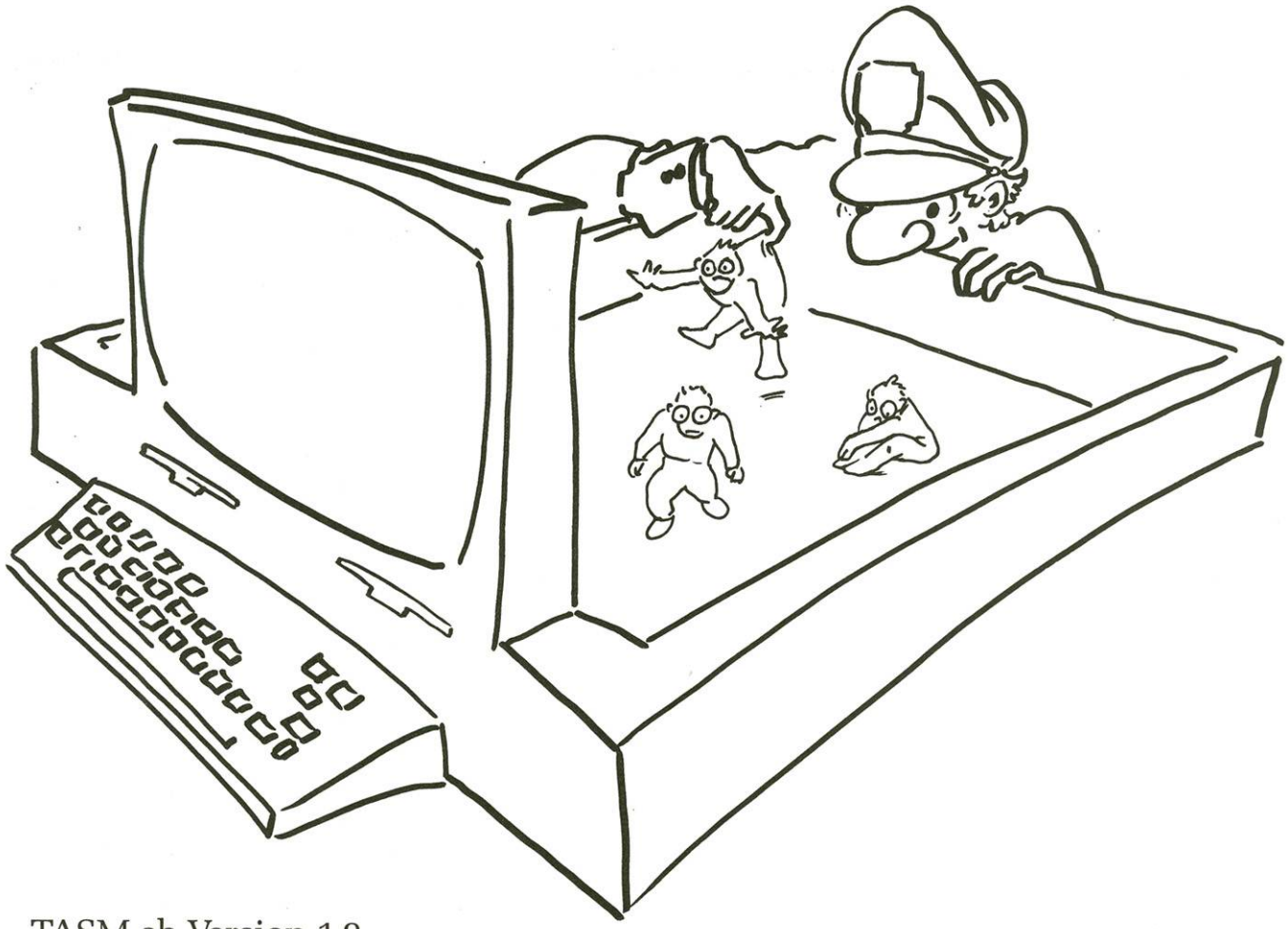
Straße, Hausnummer

Postleitzahl, Ort, Zustellpostamt

GREENPEACE

Greenpeace e.V., Vorsetzen 53, 2000 Hamburg 11
Spendenkonto: Nr. 2061-206, PGiroA Hamburg, BLZ 20010020

Z00020



TASM ab Version 1.0

Der TSR-Wächter

Verschiedene Anwendungsgebiete machen den Einsatz von speicherresidenten Programmen unumgänglich. Doch erst einmal aktiviert, wird man sie häufig ohne Neustart des Systems nicht mehr los. Beläßt man die TSR-Programme im Speicher, wird wertvoller Hauptspeicher blockiert, oder es ergeben sich sogar Unverträglichkeiten mit anderen Programmen. Abhilfe schafft in all diesen Fällen »vekdat«, der TSR-Wächter.

Speicherresidente (TSR-)Programme bleiben nach ihrem Aufruf im Arbeitsspeicher und üben dort Kontrollfunktionen aus oder warten auf ihren »Einsatz«, der entweder durch einen Interrupt oder durch einen Hotkey (spezieller Aktivierungstastendruck) ausgelöst wird. Das Betriebssystem lädt beim Systemstart Programme, beispielsweise die Gerätetreiber »print«, »mode« oder »share«, die Interrupt-Vektoren verändern. Zusätzlich können Programme wie zum Beispiel »graphics«, der »Norton Commander«, »AFD« oder »PC-Tools« speicherresident installiert werden. Jedes TSR-Programm verändert wenigstens einen Interrupt-Vektor (»print« als Beispiel gleich neun). MS-DOS speichert die Zeiger (Interrupt-Vektoren) auf diese Interrupt-Routinen im Arbeitsspeicher ab Adresse 0000:0000hex bis 0000:03ffhex. Mit Hilfe von TSR-Programmen kann man die Effizienz des Computers steigern, sie belegen allerdings auch wertvollen Speicherplatz, der von speicherhungrigen Anwenderprogrammen dringend benötigt wird. Um diesen Speicherplatz wieder freizugeben oder Inkompatibilitäten zwischen Programmen zu verhindern, muß man entweder den Computer neu starten, oder man ruft »vekdat c« auf. Das Programm »vekdat.asm« (Listing 1) zeigt die Namen der Programme, die speicherresident installiert sind (sofern sich diese Programme dem Betriebssystem ordnungsgemäß identifiziert haben), die veränderten Interrupt-Vektoren und gibt Auskunft über die Funktion der betreffenden Interrupts. Wird ein größerer, freier Speicherbereich benötigt, löscht »vekdat« auf Wunsch alle angezeigten TSR-Programme.

»vekdat« kann mit drei Parametern aufgerufen werden: »w(rite)«, »l(ist)« oder »c(ompare)«. Der Parameter »w« bewirkt das Anlegen der Vergleichsdatei »c:\vekdat.dat«. Sofern diese Laufwerks- und Pfadangabe nicht Ihren Vorstellungen oder Anforderungen entspricht, können Sie im Assembler-Quelltext »vekdat.asm« nach Belieben eine andere Vorgabe einsetzen. Den Eintrag finden Sie beim Label »Pfad« am Anfang des Quelltextes. Alle TSR-Programme, die nach dem Programmaufruf mit Parameter »w« installiert werden, können angezeigt und gelöscht werden. »vekdat w« sollte in der Datei »autoexec.bat« nach Programmen, die nicht gelöscht werden sollen, eingetragen werden.

»vekdat l« zeigt die aktuelle Liste aller Interrupt-Vektoren, die im Arbeitsspeicher ab Adresse 0000:0000hex bis 0000:03ffhex gespeichert sind.

»vekdat c« vergleicht »c:\vekdat.dat« mit der aktuellen Interrupt-Vektorliste, zeigt die Namen geladener TSR-Programme, die alten und neuen Interrupt-Vektoren sowie die Bedeutung der zugehörigen Interrupts an. Man kann das Programm mit [Esc] beenden oder mit [L] alle nicht in der Vergleichsdatei »c:\vekdat.dat« aufgeführten speicherresidenten Programme entfernen.

(Kurt Müller/bj)

```
[WP3]    1: ; Programm: vekdat.asm
[R76]    2: ; Funktion: verwaltet TSR-Programme
[9G2]    3: ; Sprache:  TASM
[124]    4: ; Autor:    Kurt Müller
[4X4]    5: ; (c) 1991 DMV Widuch GmbH & Co. KG
[ØØØ]    6:
[HC2]    7: code     segment
[8K3]    8:      assume cs:code,ds:code
[A61]    9:      org 1ØØh
[NV2]   1Ø: start:     jmp begin
[KD3]   11: tab1          db 1Ø24 dup('-')
[VP2]   12: mcbanzahl1     dw ?
[R52]   13: info1          dw Ø
[HD3]   14: tab2          db 1Ø24 dup('+')
[YP2]   15: mcbanzahl2     dw ?
[T52]   16: info2          dw Ø
[LN4]   17: pfad           db 'C:\VEKDAT.DAT',Ø
[PV2]   18: altintadr      dw ?,?
[XM3]   19: speicher1      db 8Ø dup(Ø)
[D13]   2Ø: speicherpos      dw Ø
[2F2]   21: pespeadr       dw Ø
[YC2]   22: puffer           db 3
[2L1]   23:             db 3 dup(?)
[562]   24: zeile          db Ø
[KP2]   25: gesamt           dw Ø
[FD3]   26: ausgabezaehler   db Ø
[PN2]   27: zaehler          db Ø
[MW2]   28: charzaehler      db Ø
[D42]   29: flagc           db 1
[FP2]   3Ø: kennung          db Ø
[ØØ2]   31: mcbs           db Ø
[LC2]   32: handle           dw Ø
[TE2]   33: merker           dw Ø
[SM2]   34: comflag          db Ø
[MP2]   35: mcbzaehler     dw Ø
[8B2]   36: gefunden       db Ø
[WK2]   37: mcbflag          db Ø
[3N2]   38: radierflag     db Ø
[TC2]   39: siflag           dw Ø
[VM2]   4Ø: haupt3flag     db Ø
[VE2]   41: neuadr           dw Ø
[KM2]   42: vzaehler       dw -1
[T23]   43: schlzaehler      dw -1
[GS2]   44: vekzaehler     dw Ø
[5K3]   45: vektorzaehler      dw -1
[RZ5]   46: text       db Ødh,'Weiter: [Enter]',Øffh
[854]   47: zugabe       db 'Int-Nr:',Øffh
[HS3]   48: frei       db 8Ø dup(' '),Øffh
[983]   49: trennung db Ødh,Øah
[LH2]   5Ø:         db 8Ø dup('-'),Øffh
[1A3]   51: vektor       db 32 dup('+')
[UR5]   52: alt db Øah,Ødh,'     Alter Interrupt: '
[W31]   53: db Øffh
[FT5]   54: neu db '     Neue Interrupt-Adr: ',Øffh
[8D4]   55: gibaus    db Øah,Ødh,8Ø dup('-')
[874]   56: db '      Weiter: [Enter] ','$'
[AC6]   57: neuprog  db Øah,Ødh,'      Neue Int-Adr zei'
[N34]   58: db 'gt auf Programm: ',Øffh
[VC6]   59: neuprog1 db Øah,Ødh,'      Neue Int-Adr zei'
[RG4]   6Ø: db 'gt auf Datenbereich ',Øffh
[PD5]   61: meldung      db Øah,Ødh,'(c) 1991 DOS '
[B34]   62: db 'International',Øah,Ødh
[NC5]   63: db Øah,Ødh,'Aufruf: vekdat [w]rit'
[G25]   64: db 'e, [l]ist, [c]ompare ',Øah,Ødh
[NP1]   65: db 8Ø dup('-')
[ZE5]   66: db           '         vekdat w => Erstellen '
[8N6]   67: db 'der Vergleichsdatei <c:\vekdat.dat>'
[SC1]   68: db Øah,Ødh
[ØF5]   69: db           '         vekdat l => Anzeige de'
[C85]   7Ø: db 'r Interruptvektoren',Øah,Ødh
[VF5]   71: db           '         vekdat c => Anzeige ve'
[VH6]   72: db 'ränderter Interruptvektoren und'
[SC1]   73: db Øah,Ødh
[9L5]   74: db '                          Möglichkeit, TSR-P'
[AH5]   75: db 'rogramme zu entfernen',Øah,Ødh
[SD4]   76: meldunga     db 8Ø dup ('-'),Øffh
[536]   77: meldung1     db Øah,Ødh,'Datei c:\vekda'
[DV4]   78: db 't.dat erstellt ',Øah,Ødh,Øffh
[MØ6]   79: meldung2     db Øah,Ødh,' Vekdat.dat ni'
[C94]   8Ø: db 'cht vorhanden ',Øah,Ødh
[UZ5]   81: db ' <vekdat w> aufrufen ! ',Øah,Ødh,Øffh
[AR5]   82: meldung3 db '   freigegebener MCB     '
[WN1]   83: db Ødh,Øffh
[176]   84: meldung4 db '   gelöschter INT-Vektor '
[WN1]   85: db Ødh,Øffh
[H53]   86: meldung5 db Øah,Ødh
[NP1]   87:          db 8Ø dup('-')
[ZW6]   88: meldung5a db '     Anzahl veränderter Inte'
[R64]   89: db 'rrupt-Vektoren: ',Øffh
[6G3]   9Ø: meldung6  db 8Ø dup('-')
[B26]   91: db '     Alle TSR-Programme löschen ?'
[SC1]   92: db Øah,Ødh
[DZ4]   93: db '          [esc] = abbrechen       '
[ZA5]   94: db '              [l] = löschen ',Øah,Ødh
[LH2]   95: db 8Ø dup('-'),Øffh
[RL4]   96: meldung7 db Øah,Ødh,8Ø dup('-')
[Y27]   97: db '     Anzahl zurückgeschriebener Vektor'
[DX1]   98: db 'en: ',Øffh
[TR5]   99: meldung8  db Ødh,Øah,'     Int.Nr.: ',Øffh
[5D6]  1ØØ: meldung9  db Ødh,Øah,'     Entferntes Prog'
[B82]  1Ø1: db 'ramm: ',Øffh
```

```
[SG3]  1Ø2: meldung1Ø db 8Ø dup('*')
[MM6]  1Ø3: db ' Zweiter Command.com geladen. -- Mit "'
[H17]  1Ø4: db 'exit" entfernen und vekdat c neu start'
[W52]  1Ø5: db 'en.',Ødh,Øah
[FB5]  1Ø6: db ' Programm abgebrochen ! ',Ødh,Øah
[GH2]  1Ø7: db 8Ø dup('*'),Øffh
[3H3]  1Ø8: meldung11 db 8Ø dup('-')
[Z57]  1Ø9: db '     Liste veränderter Interruptvektor'
[C72]  11Ø: db 'en: ',Øah,Ødh
[LH2]  111: db 8Ø dup('-'),Øffh
[S57]  112: meldung12 db 'freier Arbeitsspeicher(in By'
[F82]  113: db 'tes): ',Øffh
[RC6]  114: falschmsg db '       Falscher Videomodus !'
[SC1]  115: db Øah,Ødh
[B26]  116: db '     Erlaubte Modi: 2, 3 und 7',Øah,Ødh
[496]  117: db '8Øx25/monochrom und 8Øx25/16 Farben'
[P72]  118: db Øah,Ødh,Øffh
[H17]  119: intØ db '      Prozessor-Interrupt:  Divisi'
[LS4]  12Ø: db 'on durch Null                    '
[6Y2]  121: db '       ',Ødh,Øah,Øffh
[KØ7]  122: int1 db '      Prozessor-Interrupt:  Einzel'
[EY4]  123: db 'schrittmodus                     '
[6Y2]  124: db '       ',Ødh,Øah,Øffh
[1W6]  125: int2 db '      Prozessor-Interrupt:  Nicht '
[4S7]  126: db 'maskierbarer Interrrupt (Paritätsfehle'
[523]  127: db 'r)     ',Ødh,Øah,Øffh
[VW6]  128: int3 db '      Prozessor-Interrupt:  Break-'
[UE4]  129: db 'Routine                          '
[6Y2]  13Ø: db '       ',Ødh,Øah,Øffh
[B37]  131: int4 db '      Prozessor-Interrupt:  Überla'
[DU3]  132: db 'uf                               '
[6Y2]  133: db '       ',Ødh,Øah,Øffh
[LZ6]  134: int5 db '      Prozessor-Interrupt:  Bildsc'
[UF6]  135: db 'hirm-Hardcopy an Drucker senden  '
[6Y2]  136: db '       ',Ødh,Øah,Øffh
[W27]  137: int6 db '      Prozessor-Interrupt:  reserv'
[E14]  138: db 'iert                             '
[6Y2]  139: db '       ',Ødh,Øah,Øffh
[X27]  14Ø: int7 db '      Prozessor-Interrupt:  reserv'
[E14]  141: db 'iert                             '
[6Y2]  142: db '       ',Ødh,Øah,Øffh
[FW6]  143: int8 db '      Interrupt-Controller: (IRQØ)'
[G65]  144: db ' Timer-Interrupt                 '
[6Y2]  145: db '       ',Ødh,Øah,Øffh
[KW6]  146: int9 db '      Interrupt-Controller: (IRQ1)'
[CM5]  147: db ' Tastatur-Interrupt              '
[6Y2]  148: db '       ',Ødh,Øah,Øffh
[9K6]  149: intØa   db '      Interrupt-Controller: (IR'
[576]  15Ø: db 'Q2) Interrupt durch EGA-Karte    '
[PA3]  151: db '           ',Ødh,Øah,Øffh
[BK6]  152: intØb   db '      Interrupt-Controller: (IR'
[1X5]  153: db 'Q3) Interrupt durch <com2>       '
[PA3]  154: db '           ',Ødh,Øah,Øffh
[DK6]  155: intØc   db '      Interrupt-Controller: (IR'
[2X5]  156: db 'Q4) Interrupt durch <com1>       '
[PA3]  157: db '           ',Ødh,Øah,Øffh
[FK6]  158: intØd   db '      Interrupt-Controller: (IR'
[M67]  159: db 'Q5) Interrupt durch den Festplatten-Co'
[774]  16Ø: db 'ntroller  ',Ødh,Øah,Øffh
[HK6]  161: intØe   db '      Interrupt-Controller: (IR'
[W67]  162: db 'Q6) Interrupt durch den Disketten-Cont'
[3Ø4]  163: db 'roller    ',Ødh,Øah,Øffh
[LK6]  164: intØf   db '      Interrupt-Controller: (IR'
[PL6]  165: db 'Q7) Interrupt durch den Drucker   '
[PA3]  166: db '           ',Ødh,Øah,Øffh
[SK6]  167: int1Ø   db '      BIOS-Interrupt: Aufruf ei'
[GD5]  168: db 'ner Video-Funktion               '
[PA3]  169: db '           ',Ødh,Øah,Øffh
[SP6]  17Ø: int11   db '      BIOS-Interrupt: Ermittelt'
[3T5]  171: db ' Systemkonfiguration             '
[PA3]  172: db '           ',Ødh,Øah,Øffh
[UP6]  173: int12   db '      BIOS-Interrupt: Ermittelt'
[W77]  174: db ' Größe des Arbeitsspeichers (ohne ext.'
[N14]  175: db 'memory)   ',Ødh,Øah,Øffh
[YK6]  176: int13   db '      BIOS-Interrupt: Aufruf ei'
[9C7]  177: db 'ner Disketten- oder Festplattenfunktio'
[KD3]  178: db 'n          ',Ødh,Øah,Øffh
[ØL6]  179: int14   db '      BIOS-Interrupt: Aufruf ei'
[8K7]  18Ø: db 'ner Funktion für die serielle Schnitts'
[LU3]  181: db 'telle      ',Ødh,Øah,Øffh
[2L6]  182: int15   db '      BIOS-Interrupt: Aufruf ei'
[ZR6]  183: db 'ner Kassettenrecorder-Funktion   '
[PA3]  184: db '           ',Ødh,Øah,Øffh
[4L6]  185: int16   db '      BIOS-Interrupt: Aufruf ei'
[2V5]  186: db 'ner Tastatur-Funktion            '
[PA3]  187: db '           ',Ødh,Øah,Øffh
[6L6]  188: int17   db '      BIOS-Interrupt: Aufruf ei'
[ES5]  189: db 'ner Drucker-Funktion             '
[PA3]  19Ø: db '           ',Ødh,Øah,Øffh
[WH6]  191: int18   db '      BIOS-Interrupt: Start des'
[7F5]  192: db ' ROM-Kassetten-Basic             '
[PA3]  193: db '           ',Ødh,Øah,Øffh
[ZN6]  194: int19   db '      BIOS-Interrupt: Lädt und '
[7M5]  195: db 'ruft Bootsektor auf              '
[PA3]  196: db '           ',Ødh,Øah,Øffh
[UL6]  197: int1a   db '      BIOS-Interrupt: Aufruf ei'
[SZ5]  198: db 'ner Systemuhr-Funktion           '
[PA3]  199: db '           ',Ødh,Øah,Øffh
[FL6]  2ØØ: int1b   db '      Anwender-Interrupt: Abbru'
[EB7]  2Ø1: db 'chanforderung über die Tastatur (Ctrl-'
[5Z3]  2Ø2: db 'C/Break)  ',Ødh,Øah,Øffh
```

```
[9M6]  2Ø3: int1c   db '     Anwender-Interrupt: Ticke'
[9V4]  2Ø4: db 'r-Interrupt                          '
[PA3]  2Ø5: db '          ',Ødh,Øah,Øffh
[7K6]  2Ø6: int1d   db '     BIOS-Parameter: Zeiger au'
[RC7]  2Ø7: db 'f Parameter-Tabelle für den Video-Cont'
[3Ø4]  2Ø8: db 'roller    ',Ødh,Øah,Øffh
[9K6]  2Ø9: int1e   db '     BIOS-Parameter: Zeiger au'
[ZG7]  21Ø: db 'f Parameter-Tabelle für Disketten-Cont'
[3Ø4]  211: db 'roller    ',Ødh,Øah,Øffh
[BK6]  212: int1f   db '     BIOS-Parameter: Zeiger au'
[WK6]  213: db 'f Grafikzeichensatz (ASCII 128 bis 255'
[8B3]  214: db ')         ',Ødh,Øah,Øffh
[ØL6]  215: int2Ø   db '     MS-DOS-Interrupt: Beendig'
[TB6]  216: db 'ung des laufenden Programms          '
[PA3]  217: db '          ',Ødh,Øah,Øffh
[AH6]  218: int21   db '     MS-DOS-Interrupt: Aufruf '
[EA5]  219: db 'einer DOS-Funktion                   '
[PA3]  22Ø: db '          ',Ødh,Øah,Øffh
[1M6]  221: int22   db '     MS-DOS-Interrupt: Adresse'
[YR6]  222: db ', Aufruf bei Beendig. des laufenden Pr'
[Z54]  223: db 'ogramms   ',Ødh,Øah,Øffh
[3M6]  224: int23   db '     MS-DOS-Interrupt: Adresse'
[1X6]  225: db ', zu der bei [Ctrl-C] verzweigt wird '
[PA3]  226: db '          ',Ødh,Øah,Øffh
[5M6]  227: int24   db '     MS-DOS-Interrupt: Adresse'
[F47]  228: db ', zu der bei kritischen Fehlern verzwe'
[264]  229: db 'igt wird  ',Ødh,Øah,Øffh
[GP6]  23Ø: int25   db '     MS-DOS-Interrupt: Diskett'
[KY5]  231: db 'ensektor wahlfrei lesen              '
[PA3]  232: db '          ',Ødh,Øah,Øffh
[KP6]  233: int26   db '     MS-DOS-Interrupt: Diskett'
[DB6]  234: db 'ensektor wahlfrei schreiben          '
[PA3]  235: db '          ',Ødh,Øah,Øffh
[8N6]  236: int27   db '     MS-DOS-Interrupt: Program'
[616]  237: db 'm speicherresident beenden           '
[PA3]  238: db '          ',Ødh,Øah,Øffh
[PE6]  239: int28   db '     MS-DOS-Interrupt: Bei Auf'
[TN7]  24Ø: db 'ruf: DOS-Funktionen größer ØCh möglich'
[PA3]  241: db '          ',Ødh,Øah,Øffh
[526]  242: ausnahmeØ     db '     Multiplex-Interru'
[DV3]  243: db 'pt                                   '
[PA3]  244: db '          ',Ødh,Øah,Øffh
[MV5]  245: ausnahme1     db '     Für MS-DOS reserv'
[E14]  246: db 'iert                                 '
[PY3]  247: db '                    ',Ødh,Øah,Øffh
[E36]  248: ausnahme11      db '     Maus-Interrupt   '
[HL3]  249: db '                                     '
[PY3]  25Ø: db '                    ',Ødh,Øah,Øffh
[TW5]  251: ausnahme2     db '     Für Erweiterung r'
[FH4]  252: db 'eserviert                            '
[PY3]  253: db '                    ',Ødh,Øah,Øffh
[LP5]  254: ausnahme3     db '     Zeiger auf Parame'
[V87]  255: db 'tertabelle für Festplatten-Controller '
[PY3]  256: db '                    ',Ødh,Øah,Øffh
[D16]  257: ausnahme4     db '     Für Erweiterungen'
[ZU4]  258: db ' reserviert                          '
[PY3]  259: db '                    ',Ødh,Øah,Øffh
[YØ6]  26Ø: ausnahme5     db '     Für Anwender-Inte'
[MA5]  261: db 'rrupt reserviert                     '
[PY3]  262: db '                    ',Ødh,Øah,Øffh
[LR5]  263: ausnahme6     db '     Nicht verwendet  '
[HL3]  264: db '                                     '
[PY3]  265: db '                    ',Ødh,Øah,Øffh
[3Z5]  266: ausnahme7     db '     Für Basic reservi'
[114]  267: db 'ert                                  '
[PY3]  268: db '                    ',Ødh,Øah,Øffh
[PZ5]  269: ausnahme8     db '     Basic-Interpreter'
[BP4]  27Ø: db '-Interrupt                           '
[PY3]  271: db '                    ',Ødh,Øah,Øffh
[TR5]  272: ausnahme9     db '     Nicht verwendet  '
[HL3]  273: db '                                     '
[PY3]  274: db '                    ',Ødh,Øah,Øffh
[UU5]  275: ausnahme1Ø      db '     EMS-Interrupt    '
[HL3]  276: db '                                     '
[PY3]  277: db '                    ',Ødh,Øah,Øffh
[ZP6]  278: ;überprüft die Kommandozeile,setzt die
[326]  279: ;Kennung und springt zum Verteiler
[EW2]  28Ø: begin:     call bild
[EU1]  281:       jnc gonext
[4D1]  282:       jmp ende
[C23]  283: gonext: mov si,8Øh
[DV3]  284:       cmp byte ptr ds:[si],Ø
[EF1]  285:       jne case1
[2X1]  286:       jmp syntax
[X52]  287: case1:      inc si
[911]  288:       inc si
[322]  289:       inc kennung
[954]  29Ø:       cmp byte ptr ds:[si],'1'
[GF1]  291:       jne case2
[MØ2]  292:       jmp weiter1
[R43]  293: case2:      inc kennung
[N64]  294:       cmp byte ptr ds:[si],'w'
[KF1]  295:       jne case3
[MØ2]  296:       jmp weiter1
[T43]  297: case3:      inc kennung
[544]  298:       cmp byte ptr ds:[si],'c'
[BL1]  299:       je weiter1
[2X1]  3ØØ:       jmp syntax
[LB1]  3Ø1: weiter1:
[BC1]  3Ø2:       push ds
[8C1]  3Ø3:       push cs
[G31]  3Ø4:       pop es
[RE5]  3Ø5: ;kopiert Ram von Ø:Ø bis ØØØØ:Ø3ffh
[S45]  3Ø6: ;(Interruptvektoren) nach tab1
[YR1]  3Ø7:       lea di,tab1
[GH1]  3Ø8:       xor ax,ax
[1H1]  3Ø9:       mov ds,ax
[VG1]  31Ø:       xor si,si
[AL1]  311:       mov cx,512
[VX4]  312: ;Ø bis 3ffh = 1Ø24 bytes = 512 words
[T42]  313:       repnz movsw
[E31]  314:       pop ds
[CØ7]  315: ;Verteiler ruft die Hauptprozeduren auf
[P72]  316:       cmp kennung,1
[BYØ]  317:       jne c1
[PW1]  318:       call haupt1
[4D1]  319:       jmp ende
[T23]  32Ø: c1:      cmp kennung,2
[EYØ]  321:       jne c2
[UW1]  322:       call haupt2
[4D1]  323:       jmp ende
[Y23]  324: c2:      cmp kennung,3
[VB1]  325:       jne ende
[2K2]  326:       lea si,meldung11
[LK2]  327:       call writetext
[YW1]  328:       call haupt3
[D12]  329:       cmp gesamt,Ø
[451]  33Ø:       je ende
[2T6]  331: ;sucht mit DOS-Funktion 4eh (finde ersten
[BK4]  332: ;Eintrag) nach c:\vekdat.dat
[V57]  333: ;vorhanden => ok, sonst 'vekdat w' aufrufen
[ZM1]  334:       mov ah,4eh
[BN1]  335:       mov cx,22h
[UV1]  336:       lea dx,pfad
[381]  337:       int 21h
[ØM1]  338:       jc vorende
[V91]  339:       mov ah,2
[GV1]  34Ø:       mov dx,14ØØh
[Y71]  341:       int 1Øh
[5H2]  342:       lea si,meldung6
[LK2]  343:       call writetext
[ØC5]  344: ;Abfrage : löschen oder abbrechen
[5B1]  345: endlos1:
[ØF1]  346:       xor ah,ah
[G81]  347:       int 16h
[VW1]  348:       cmp ax,266ch
[SU3]  349: ;Scan-/Asciicode von '1'
[8E1]  35Ø:       jne zwei
[3E1]  351:       call bild
[2X1]  352:       call haupt4
[4D1]  353:       jmp ende
[SØ3]  354: zwei:      cmp ax,Ø11bh
[9K4]  355: ;Scan-/Asciicode von 'enter'
[XY1]  356:       jne endlos1
[W71]  357:       je endex
[W24]  358: syntax:     lea si,meldung
[LK2]  359:       call writetext
[4D1]  36Ø:       jmp ende
[WE1]  361: vorende:
[NG2]  362:       lea si,meldung2
[LK2]  363:       call writetext
[ØS2]  364: ende:      mov ah,4ch
[381]  365:       int 21h
[LX2]  366: endex:      call bild
[VM1]  367:       mov ah,4ch
[381]  368:       int 21h
[AZ3]  369: ;                                ( vekdat 1 )
[WR6]  37Ø: ;gibt die aktuelle Liste aller Interrupt-
[M72]  371: ;vektoren aus
[AC3]  372: haupt1          proc near
[HB1]  373:       mov cx,4
[464]  374: ;4 Anzeigen in einer Zeile
[H54]  375: sprung: inc schlzaehler
[542]  376:       inc vzaehler
[532]  377:       lea si,zugabe
[LK2]  378:       call writetext
[HS2]  379:       mov ax,vzaehler
[792]  38Ø:       call writehex
[4Z1]  381:       mov ax,Øe3dh
[Y71]  382:       int 1Øh
[WC1]  383:       push cx
[8S2]  384:       mov si,vzaehler
[BA1]  385:       mov cl,2
[UE1]  386:       shl si,cl
[XW2]  387:       add si,offset tab1
[V31]  388:       pop cx
[XS2]  389:       call writedword
[HW1]  39Ø:       mov ax,Øe2Øh
[Y71]  391:       int 1Øh
[ØW2]  392:       cmp vzaehler,Øffh
[D17]  393: ;256 Int-Vektoren zu je 2 words ausgegeben
[BX2]  394: ;insgesamt 1KByte
[451]  395:       je ende
[C42]  396:       loop sprung
[S23]  397:       cmp schlzaehler,9Ø
[E36]  398: ;Bildschirm voll,auf 'enter' warten
[2H1]  399:       jae warte
[HB1]  4ØØ:       mov cx,4
[ZV1]  4Ø1:       jmp sprung
[RM3]  4Ø2: warte:      call carriage
```

```
[KZ1]  4Ø3:       lea si,text
[LK2]  4Ø4:       call writetext
[HG1]  4Ø5: schleife:
[ØF1]  4Ø6:       xor ah,ah
[G81]  4Ø7:       int 16h
[6X1]  4Ø8:       cmp ax,1cØdh
[F22]  4Ø9:       jne schleife
[8Z2]  41Ø:       mov schlzaehler,Ø
[3E1]  411:       call bild
[7T6]  412: ;Bildschirm löschen für neue Anzeige
[HB1]  413:       mov cx,4
[ZV1]  414:       jmp sprung
[3PØ]  415:       ret
[6H2]  416: haupt1          endp
[ØØØ]  417:
[BN3]  418: ;                          ( vekdat w )
[KZ6]  419: ;erstellt vekdat.dat u. schreibt die Liste
[RW5]  42Ø: ;der Int-Vekt,die Anzahl der vorher
[K97]  421: ;installierten TSR-Progs und info1 = freier
[UD5]  422: ;Arbeitsspeicher nach vekdat.dat
[DC3]  423: haupt2          proc near
[MY1]  424:       call unter1
[UM1]  425:       mov ah,3ch
[WH6]  426: ;DOS-Funktion 3ch(erstelle Handle-Datei)
[UV1]  427:       lea dx,pfad
[3M1]  428:       mov cl,2Øh
[381]  429:       int 21h
[P62]  43Ø:       mov handle,ax
[NL1]  431:       mov ah,4Øh
[RZ6]  432: ;DOS-Funktion 4Øh(schreibe in Handle-Datei)
[N62]  433:       mov bx,handle
[4U1]  434:       mov cx,1Ø28
[BT1]  435:       lea dx,tab1
[381]  436:       int 21h
[YM1]  437:       mov ah,3eh
[717]  438: ;DOS-Funktion 3eh (schließe Handle-Datei)
[N62]  439:       mov bx,handle
[381]  44Ø:       int 21h
[HG2]  441:       lea si,meldung1
[LK2]  442:       call writetext
[3PØ]  443:       ret
[9H2]  444: haupt2          endp
[ØØØ]  445:
[987]  446: ;benutzt undokumentierte Funktion 52h zur
[P77]  447: ;Ermittlung des ersten MCB (memory control
[2M6]  448: ;block), zählt die TSR-Progs, die vor dem
[8Ø7]  449: ;aufruf von 'vekdat w' installiert wurden
[SD3]  45Ø: unter1          proc near
[EC1]  451:       push es
[BC1]  452:       push ds
[TC1]  453:       push bx
[UL1]  454:       mov ah,52h
[381]  455:       int 21h
[AT3]  456: ;Adresse 1.MCB in ES:BX-2
[FØ1]  457:       dec bx
[FØ1]  458:       dec bx
[R84]  459:       mov es,word ptr es:[bx]
[6T1]  46Ø: naechster:
[FØ4]  461:       cmp byte ptr es:[Ø],'M'
[TF6]  462: ;MCB-Kennung = 'M'=>Nachfolger vorhanden
[P55]  463: ;MCB-Kennung = 'Z'=>letzter MCB
[242]  464:       jne letzter
[4N3]  465:       cmp word ptr es:[1],Ø
[5M5]  466: ;MCB-offset 1 = Ø => MCB freigegeben
[E71]  467:       je ueber
[6H1]  468:       mov bx,es
[RW3]  469:       mov ax,word ptr es:[1]
[DØ1]  47Ø:       dec ax
[TF1]  471:       cmp ax,bx
[WG1]  472:       jne ueber
[AD2]  473:       inc mcbzaehler
[R25]  474: ueber:  mov cx,word ptr es:[3]
[U47]  475: ;MCB-offset 3 = Abstand zum nächsten MCB-1
[P11]  476:       inc cx
[4H1]  477:       mov ax,es
[BD1]  478:       add ax,cx
[M66]  479: ;ax=Adresse alter MCB,cx=Abstand zum
[GF2]  48Ø: ;nächsten MCB
[3H1]  481:       mov es,ax
[LA2]  482:       jmp naechster
[AG1]  483: letzter:
[KM5]  484:       mov ax,word ptr ds:[mcbzaehler]
[DV2]  485:       mov mcbanzahl1,ax
[XW3]  486:       mov bx,word ptr es:[3]
[E77]  487: ;letzter MCB-offset 3 = freier Arbeitsspei-
[HZ4]  488: ;cher in Paragraphen (16 bytes)
[812]  489:       mov info1,bx
[T31]  49Ø:       pop bx
[E31]  491:       pop ds
[G31]  492:       pop es
[3PØ]  493:       ret
[MK2]  494: unter1          endp
[ØØØ]  495:
[4Z3]  496: ;                          ( vekdat c )
[VM6]  497: ;vergleicht tab2 (Liste aus vekdat.dat)
[5Y5]  498: ;mit tab1 (aktuelle Int-Vekt-Liste),
[R46]  499: ;überspringt dabei die TSR-Progs, die
[GL6]  5ØØ: ;vor Aufruf von 'vekdat w' installiert
[Z67]  5Ø1: ;wurden;werden Unterschiede festgestellt,
[4U5]  5Ø2: ;ruft Haupt3 die proc 'anzeige' auf
```

```
[GC3]  5Ø3: haupt3          proc near
[ØB1]  5Ø4:       call dos
[FY1]  5Ø5:       cmp ax,7f7fh
[VD1]  5Ø6:       je finito
[YR1]  5Ø7:       lea di,tab1
[ØT1]  5Ø8:       lea si,tab2
[VL1]  5Ø9:       mov cx,256
[7U6]  51Ø: retour:     mov ax,word ptr ds:[si] ;tab2
[VZ4]  511:       cmp ax,word ptr ds:[di] ;tab1
[3V4]  512: ;vergleicht Offset-Adressen
[ZE1]  513:       jne fall1
[N94]  514:       mov ax,word ptr ds:[si+2]
[574]  515:       cmp ax,word ptr ds:[di+2]
[615]  516: ;vergleicht Segment-Adressen
[ZE1]  517:       jne fall1
[371]  518:       add si,4
[561]  519:       add di,4
[M42]  52Ø:       loop retour
[V91]  521:       mov ah,2
[9V1]  522:       mov dx,Ø1ØØh
[Y71]  523:       int 1Øh
[2L2]  524:       lea si,meldung5a
[LK2]  525:       call writetext
[WC1]  526:       push cx
[ZC1]  527:       push dx
[282]  528:       mov dx,gesamt
[PH1]  529:       xor cx,cx
[NY2]  53Ø:       call writedecimal
[X31]  531:       pop dx
[V31]  532:       pop cx
[TN2]  533:       lea si,trennung
[LK2]  534:       call writetext
[SU1]  535:       jmp finito
[RE3]  536: fall1:      mov siflag,si
[8T1]  537:       inc gesamt
[3B1]  538:       push di
[ZM2]  539:       lea di,speicher1
[823]  54Ø:       add di,speicherpos
[YG1]  541:       mov ax,si
[ZYØ]  542:       stosw
[ST2]  543:       inc speicherpos
[ST2]  544:       inc speicherpos
[G21]  545:       pop di
[DØ2]  546:       call ausgabe
[371]  547:       add si,4
[561]  548:       add di,4
[HØ1]  549:       dec cx
[V91]  55Ø:       cmp cx,Ø
[GE1]  551:       jl finito
[DW1]  552:       jmp retour
[MC4]  553: finito: cmp cs:[gesamt],Ø
[P41]  554:       je pos3
[MW1]  555:       mov dx,1527h
[6U1]  556:       jmp endpos
[323]  557: pos3:       mov dx,Ø327h
[P43]  558: endpos:     xor bh,bh
[V91]  559:       mov ah,2
[Y71]  56Ø:       int 1Øh
[3K2]  561:       lea si,meldung12
[LK2]  562:       call writetext
[MY1]  563:       call unter1
[W72]  564:       call rechnung
[3PØ]  565:       ret
[CH2]  566: haupt3          endp
[ØØØ]  567:
[SZ6]  568: ;multipliziert dx (freier Arbeitsspeicher
[RV3]  569: ;in Paragraphen) mit 16
[3E3]  57Ø: rechnung      proc near
[WC1]  571:       push cx
[ZC1]  572:       push dx
[PH1]  573:       xor cx,cx
[332]  574:       mov dx,info1
[Z91]  575:       shl dx,1
[C91]  576:       rcl cx,1
[Z91]  577:       shl dx,1
[C91]  578:       rcl cx,1
[Z91]  579:       shl dx,1
[C91]  58Ø:       rcl cx,1
[Z91]  581:       shl dx,1
[C91]  582:       rcl cx,1
[NY2]  583:       call writedecimal
[X31]  584:       pop dx
[V31]  585:       pop cx
[3PØ]  586:       ret
[CK2]  587: rechnung      endp
[ØØØ]  588:
[PØ6]  589: ;ermittelt aus den MCBs die Namen der
[AZ6]  59Ø: ;TSRs,deren PSP-Adresse haupt3 übergibt
[2B6]  591: ;und zeigt die Namen,die alten und neuen
[4T3]  592: ;Interruptvektoren an
[PP3]  593: anzeige          proc near
[PC1]  594:       push ax
[GC1]  595:       push si
[WC1]  596:       push cx
[TC1]  597:       push bx
[YF2]  598:       inc cs:[zeile]
[4H1]  599:       mov ax,es
[DØ1]  6ØØ:       dec ax
[3H1]  6Ø1:       mov es,ax
[8B1]  6Ø2:       mov si,8
```

```
[D87]  6Ø3: ;MCB-offset 8 = Name des zugehörigen Prog.
[NB1]  6Ø4:       mov cx,8
[L73]  6Ø5: ;Max.-Länge 8 bytes
[VM1]  6Ø6:       mov ah,Øeh
[HA1]  6Ø7: circle1:
[S74]  6Ø8:       mov al,byte ptr es:[si]
[V34]  6Ø9:       or al,al ;vergleicht mit Ø
[VH6]  61Ø: ;wenn kürzer als 8 bytes,'Ø' als Endmarke
[YF1]  611:       je exitus
[Y71]  612:       int 1Øh
[MT6]  613: ;Ausgabe: Funktion Øeh,Bios-Interrupt 1Øh
[911]  614:       inc si
[Z12]  615:       loop circle1
[BF2]  616: exitus: inc cx
[P11]  617:       inc cx
[P11]  618:       inc cx
[HW1]  619:       mov ax,Øe2Øh
[U72]  62Ø: abst:      int 1Øh
[RG1]  621:       loop abst
[GP2]  622:       lea si,altintadr
[XS2]  623:       call writedword
[MM2]  624:       cmp altintadr,Ø
[AU1]  625:       jne pspadr
[TG2]  626:       lea si,meldung3
[LK2]  627:       call writetext
[CB1]  628:       jmp car
[X74]  629: pspadr:      lea si,meldung4
[LK2]  63Ø:       call writetext
[583]  631: car:      call carriage
[4V2]  632:       cmp cs:[zeile],1Ø
[YZ6]  633: ;nach 9 Ausgaben anhalten,auf enter warten
[A12]  634:       jb carweiter
[932]  635:       lea dx,gibaus
[2A1]  636:       mov ah,9
[381]  637:       int 21h
[TE1]  638: endless:
[ØF1]  639:       xor ah,ah
[G81]  64Ø:       int 16h
[6X1]  641:       cmp ax,1cØdh
[H22]  642:       jne endless
[NT2]  643:       mov cs:[zeile],Ø
[3E1]  644:       call bild
[MT1]  645: carweiter:
[T31]  646:       pop bx
[V31]  647:       pop cx
[E31]  648:       pop si
[R31]  649:       pop ax
[3PØ]  65Ø:       ret
[ZU2]  651: anzeige          endp
[ØØØ]  652:
[9F4]  653: ;ruft Löschprozeduren auf
[LC3]  654: haupt4          proc near
[G12]  655:       call command
[FN2]  656:       cmp comflag,Øfh
[9L1]  657:       je abbruch
[HK2]  658:       lea si,meldunga
[LK2]  659:       call writetext
[ØB1]  66Ø:       call dos
[8W1]  661:       mov info1,Ø
[WY1]  662:       call unter3
[BY1]  663:       call delete
[FB1]  664: abbruch:
[F72]  665:       mov comflag,Ø
[3PØ]  666:       ret
[FH2]  667: haupt4          endp
[ØØØ]  668:
[N87]  669: ;überschreibt neue Int-Vektoren mit alten
[PA2]  67Ø: ;Int-Vektoren
[VC3]  671: delete          proc near
[YR1]  672:       lea di,tab1
[ØT1]  673:       lea si,tab2
[VL1]  674:       mov cx,256
[9P5]  675: rundum: mov ax,word ptr ds:[si]
[164]  676:       cmp ax,word ptr ds:[di]
[Ø86]  677: ;Vergleich alte/neue Offset-Adressen
[LE1]  678:       jne fail1
[N94]  679:       mov ax,word ptr ds:[si+2]
[574]  68Ø:       cmp ax,word ptr ds:[di+2]
[ZA6]  681: ;Vergleich alte/neue Segment-Adressen
[LE1]  682:       jne fail1
[371]  683:       add si,4
[561]  684:       add di,4
[452]  685:       loop rundum
[FU1]  686:       jmp walkon
[U75]  687: fail1:      mov ax,word ptr ds:[di+2]
[143]  688:       mov altintadr+2,ax
[3F6]  689: ;Eintrag für Anzeige Segment-Adresse
[N74]  69Ø:       mov bx,word ptr ds:[di]
[6V2]  691:       mov altintadr,bx
[KE6]  692: ;Eintrag für Anzeige Offset-Adresse
[3H1]  693:       mov es,ax
[7M1]  694:       mov ah,49h
[381]  695:       int 21h
[P17]  696: ;DOS-Funkt. 49h(Speicherbereich freigeben)
[8V6]  697: ;ES:Segmentadresse des Speicherbereichs
[8T1]  698:       jc zeichen1
[NV1]  699:       call habit1
[3G1]  7ØØ:       jmp oben1
[6H1]  7Ø1: zeichen1:
[PR2]  7Ø2:       call exefreigabe
```

```
[KR2]  7Ø3: oben1:      mov di,si
[HY2]  7Ø4:       sub di,offset tab2
[GH1]  7Ø5:       xor ax,ax
[3H1]  7Ø6:       mov es,ax
[YR1]  7Ø7:       call intnr
[ØZØ]  7Ø8:       movsw
[YD6]  7Ø9: ;alte Offset-Adresse zurückschreiben
[ØZØ]  71Ø:       movsw
[RH6]  711: ;alte Segment-Adresse zurückschreiben
[6F2]  712:       inc vekzaehler
[HØ1]  713:       dec cx
[RF1]  714:       mov di,si
[K45]  715:       sub di,offset tab2 - offset tab1
[AR5]  716: ;Offset-Adressen wieder anpassen
[V91]  717:       cmp cx,Ø
[CE1]  718:       jl walkon
[BW1]  719:       jmp rundum
[WA1]  72Ø: walkon:
[9H2]  721:       lea si,meldung7
[LK2]  722:       call writetext
[WC1]  723:       push cx
[ZC1]  724:       push dx
[413]  725:       mov dx,vekzaehler
[PH1]  726:       xor cx,cx
[NY2]  727:       call writedecimal
[X31]  728:       pop dx
[V31]  729:       pop cx
[9F3]  73Ø:       cmp vektorzaehler,-1
[CD1]  731:       je vorbei
[DH2]  732:       lea si,meldung8
[LK2]  733:       call writetext
[4Y2]  734:       call schreibveknr
[8G3]  735: vorbei: call carriage
[3PØ]  736:       ret
[RH2]  737: delete          endp
[ØØØ]  738:
[VR7]  739: ;ermittelt die Adressen der PSP für die DOS
[ØW6]  74Ø: ;-Funktion 49h (Speicherbereich freigeben)
[C95]  741: ;DOS-Funktion 52h wie in 'unter1'
[YD3]  742: unter3          proc near
[TH1]  743:       xor dx,dx
[UL1]  744:       mov ah,52h
[381]  745:       int 21h
[FØ1]  746:       dec bx
[FØ1]  747:       dec bx
[R84]  748:       mov es,word ptr es:[bx]
[TZ1]  749: naechster3:
[FØ4]  75Ø:       cmp byte ptr es:[Ø],'M'
[1F1]  751:       jne ende3
[4N3]  752:       cmp word ptr es:[1],Ø
[MA1]  753:       je ueber3
[PC1]  754:       push ax
[6H1]  755:       mov bx,es
[RW3]  756:       mov ax,word ptr es:[1]
[DØ1]  757:       dec ax
[TF1]  758:       cmp ax,bx
[R31]  759:       pop ax
[NN1]  76Ø:       jne ueber3
[CA2]  761:       cmp gefunden,1
[NH1]  762:       jne suche
[A94]  763:       cmp word ptr es:[16],2Øcdh
[NN1]  764:       jne ueber3
[U33]  765:       mov altintadr+2,es
[6P2]  766:       mov altintadr,Ø
[M22]  767:       call filtern
[BY1]  768:       cmp ax,9d9dh
[MA1]  769:       je ueber3
[D62]  77Ø:       call radieren
[YP1]  771:       jmp ueber3
[L82]  772: suche:      inc dx
[1U2]  773:       cmp mcbanzahl2,dx
[NN1]  774:       jne ueber3
[YB2]  775:       mov gefunden,1
[6F5]  776: ueber3:      mov ax,word ptr es:[3]
[K11]  777:       inc ax
[8H1]  778:       mov cx,es
[BD1]  779:       add ax,cx
[3H1]  78Ø:       mov es,ax
[CF2]  781:       jmp naechster3
[AL3]  782: ende3:      mov gefunden,Ø
[3PØ]  783:       ret
[UK2]  784: unter3          endp
[ØØØ]  785:
[DN4]  786: ;löscht das TSR-Programm
[A24]  787: ;mit der DOS-Funktion 49h
[4C3]  788: radieren      proc near
[PC1]  789:       push ax
[ZC1]  79Ø:       push dx
[EC1]  791:       push es
[GC1]  792:       push si
[1X3]  793:       mov dx,word ptr es:[3]
[FX1]  794:       add info1,dx
[4H1]  795:       mov ax,es
[K11]  796:       inc ax
[3H1]  797:       mov es,ax
[EC1]  798:       push es
[7M1]  799:       mov ah,49h
[381]  8ØØ:       int 21h
[164]  8Ø1:       mov es,word ptr es:[2ch]
[7M1]  8Ø2:       mov ah,49h
```

```
[381]  8Ø3:       int 21h
[G31]  8Ø4:       pop es
[HH2]  8Ø5:       lea si,meldung9
[LK2]  8Ø6:       call writetext
[Ø12]  8Ø7:       call anzeige
[E31]  8Ø8:       pop si
[G31]  8Ø9:       pop es
[X31]  81Ø:       pop dx
[R31]  811:       pop ax
[3PØ]  812:       ret
[DG2]  813: radieren      endp
[ØØØ]  814:
[V74]  815: ;Hilfsprozedur von delete
[R93]  816: habit1          proc near
[GC1]  817:       push si
[HH2]  818:       lea si,meldung9
[LK2]  819:       call writetext
[Ø12]  82Ø:       call anzeige
[4H1]  821:       mov ax,es
[K11]  822:       inc ax
[3H1]  823:       mov es,ax
[E31]  824:       pop si
[3PØ]  825:       ret
[LE2]  826: habit1          endp
[ØØØ]  827:
[1V6]  828: ;überprüft,ob 2. command.com geladen ist
[9K6]  829: ;ja=>gibt Meldung aus,bricht Programm ab
[KG7]  83Ø: ;Wird aus einem Anwenderprogramm heraus die
[767]  831: ;DOS-Ebene aufgerufen,wird der Kommandopro-
[PS7]  832: ;zessor noch einmal geladen;läßt man dann
[HW5]  833: ;löschen, stürzt der Computer ab
[KN3]  834: command          proc near
[PC1]  835:       push ax
[EC1]  836:       push es
[GC1]  837:       push si
[3B1]  838:       push di
[TC1]  839:       push bx
[UL1]  84Ø:       mov ah,52h
[381]  841:       int 21h
[FØ1]  842:       dec bx
[FØ1]  843:       dec bx
[R84]  844:       mov es,word ptr es:[bx]
[CR1]  845: beginning:
[4N3]  846:       cmp word ptr es:[1],Ø
[KØ2]  847:       je notexist
[FØ4]  848:       cmp byte ptr es:[Ø],'M'
[812]  849:       jne retour1
[EC1]  85Ø:       push es
[GW3]  851:       mov es,word ptr es:[1]
[4H1]  852:       mov ax,es
[F44]  853:       cmp word ptr es:[Øch],ax
[4Ø7]  854: ;Vergleicht Segmentadresse des PSP mit der
[WA7]  855: ;Seg-Adr der Routine zum Beenden des Progra
[GZ6]  856: ;mms.Nur bei command.com sind die Adressen
[9B1]  857: ;gleich.
[G31]  858:       pop es
[LA1]  859:       je exist
[XN1]  86Ø: notexist:
[VW3]  861:       mov ax,word ptr es:[3]
[K11]  862:       inc ax
[6H1]  863:       mov bx,es
[AD1]  864:       add ax,bx
[3H1]  865:       mov es,ax
[T82]  866:       jmp beginning
[B83]  867: exist:  inc comflag
[562]  868:       cmp comflag,4
[LG6]  869: ;Dos legt für ein Programm und für sein
[6B6]  87Ø: ;Environment je einen MCB an.Der erste
[EV6]  871: ;command.com wird hier nicht geschützt,
[CR6]  872: ;er muß vor vekdat w geladen werden !
[V72]  873:       jne notexist
[1R2]  874:       mov comflag,Øfh
[1K2]  875:       lea si,meldung1Ø
[LK2]  876:       call writetext
[GD1]  877: retour1:
[T31]  878:       pop bx
[G21]  879:       pop di
[E31]  88Ø:       pop si
[G31]  881:       pop es
[R31]  882:       pop ax
[3PØ]  883:       ret
[VT2]  884: command          endp
[ØØØ]  885:
[MW5]  886: ;löscht mit der DOS-Funktion 49h
[X73]  887: ;TSR-Exe-Programme
[YX3]  888: exefreigabe     proc near
[EC1]  889:       push es
[PC1]  89Ø:       push ax
[GC1]  891:       push si
[ZC1]  892:       push dx
[4H1]  893:       mov ax,es
[4D1]  894:       sub ax,16
[3H1]  895:       mov es,ax
[GH1]  896:       mov dx,ax
[7M1]  897:       mov ah,49h
[381]  898:       int 21h
[164]  899:       mov es,word ptr es:[2ch]
[7M1]  9ØØ:       mov ah,49h
[381]  9Ø1:       int 21h
[561]  9Ø2:       jc kein
```

```
[HH2]  9Ø3:       lea si,meldung9
[LK2]  9Ø4:       call writetext
[6H1]  9Ø5:       mov es,dx
[Ø12]  9Ø6:       call anzeige
[MB1]  9Ø7:       jmp weg
[EC3]  9Ø8: kein:      mov mcbflag,1
[V92]  9Ø9: weg:      pop dx
[E31]  91Ø:       pop si
[R31]  911:       pop ax
[G31]  912:       pop es
[3PØ]  913:       ret
[U23]  914: exefreigabe      endp
[ØØØ]  915:
[CL5]  916: ;zeigt die Interrupts,die von TSR-
[AV5]  917: ;Programmen verändert worden sind
[V34]  918: schreibveknr      proc near
[KL3]  919:       mov cx,vektorzaehler
[P11]  92Ø:       inc cx
[B62]  921:       lea bx,vektor
[GH1]  922:       xor ax,ax
[TH1]  923:       xor dx,dx
[U82]  924: runde:      xlat
[792]  925:       call writehex
[W91]  926:       cmp cx,1
[AL1]  927:       je alldone
[PC1]  928:       push ax
[TC1]  929:       push bx
[XY1]  93Ø:       mov ax,Øe2ch
[Y71]  931:       int 1Øh
[T31]  932:       pop bx
[R31]  933:       pop ax
[MØ1]  934:       inc dl
[HF1]  935:       mov al,dl
[ØU1]  936:       loop runde
[GD1]  937: alldone:
[3PØ]  938:       ret
[483]  939: schreibveknr      endp
[ØØØ]  94Ø:
[VX4]  941: ;berechnet Interruptnummern
[3Y6]  942: ;Die Liste der Int-Vekt beginnt ab Offset Ø
[A57]  943: ;Um die Int.Nr zu ermitteln, muß die Offs-
[YS4]  944: ;Adr durch 4 dividiert werden.
[UY3]  945: ;z.B. Offs.= ØØ84h => int 21h
[C73]  946: intnr          proc near
[TC1]  947:       push bx
[8W1]  948:       cmp di,3fch
[M71]  949:       ja nicht
[PC1]  95Ø:       push ax
[WC1]  951:       push cx
[V23]  952:       inc vektorzaehler
[GL3]  953:       mov bx,vektorzaehler
[FG1]  954:       mov ax,di
[BA1]  955:       mov cl,2
[VF1]  956:       shr ax,cl
[SX4]  957: ;Offset-Adr dividiert durch 4
[ØS2]  958:       mov vektor+bx,al
[V31]  959:       pop cx
[R31]  96Ø:       pop ax
[H92]  961: nicht:      pop bx
[3PØ]  962:       ret
[BG2]  963: intnr          endp
[ØØØ]  964:
[MM6]  965: ;kopiert Int-Vekt-Liste von vekdat.dat
[2K1]  966: ;nach tab2
[3Ø3]  967: dos          proc near
[MW1]  968:       mov ax,3dØ2h
[776]  969: ;3dh:Handle-Datei öffnen (vekdat.dat)
[UV1]  97Ø:       lea dx,pfad
[381]  971:       int 21h
[FZ1]  972:       jc dosfehler
[LH6]  973: ;Vekdat.dat nicht vorhanden=>abbrechen
[P62]  974:       mov handle,ax
[ØN1]  975:       mov ah,3fh
[P54]  976: ;3fh:Lies aus Handle-Datei
[N62]  977:       mov bx,handle
[4U1]  978:       mov cx,1Ø28
[FT1]  979:       lea dx,tab2
[381]  98Ø:       int 21h
[YM1]  981:       mov ah,3eh
[5C4]  982: ;3eh:schließe Handle Datei
[N62]  983:       mov bx,handle
[381]  984:       int 21h
[X22]  985:       jmp dosende
[GT1]  986: dosfehler:
[NG2]  987:       lea si,meldung2
[LK2]  988:       call writetext
[1Ø2]  989:       mov ax,7f7fh
[WD1]  99Ø: dosende:
[3PØ]  991:       ret
[V62]  992: dos          endp
[ØØØ]  993:
[PE7]  994: ;prüft den Videomodus,löscht den Bildschirm
[GS4]  995: ;und setzt das Farbattribut
[F23]  996: bild          proc near
[PC1]  997:       push ax
[TC1]  998:       push bx
[WC1]  999:       push cx
[ZC1] 1ØØØ:       push dx
[EC1] 1ØØ1:       push es
[3B1] 1ØØ2:       push di
```

```
[XM1] 1Ø03:       mov ah,Øfh
[Y71] 1Ø04:       int 1Øh
[U81] 1Ø05:       cmp al,7
[M71] 1Ø06:       je mono
[P81] 1Ø07:       cmp al,3
[RVØ] 1Ø08:       je cga
[N81] 1Ø09:       cmp al,2
[RVØ] 1Ø1Ø:       je cga
[1V2] 1Ø11:       jmp falschmodus
[F43] 1Ø12: cga:      mov ax,Øb8ØØh
[5M1] 1Ø13:       mov dh,Ø7h
[4A6] 1Ø14: ;Farbattribut in dh(weiß auf schwarz)
[P21] 1Ø15:       jmp go
[X93] 1Ø16: mono:      mov ax,ØbØØØh
[5M1] 1Ø17:       mov dh,Ø7h
[6F2] 1Ø18: go:      mov es,ax
[EF1] 1Ø19:       xor di,di
[XE1] 1Ø2Ø:       mov ah,dh
[ZL1] 1Ø21:       mov al,2Øh
[BU1] 1Ø22:       mov cx,2Ø48
[852] 1Ø23:       repnz stosw
[NM1] 1Ø24:       mov ah,Øbh
[3V1] 1Ø25:       mov bx,ØØØØh
[3M4] 1Ø26: ;Rahmenfarbe in bl(schwarz)
[Y71] 1Ø27:       int 1Øh
[V91] 1Ø28:       mov ah,2
[3F1] 1Ø29:       xor bh,bh
[TH1] 1Ø3Ø:       xor dx,dx
[Y71] 1Ø31:       int 1Øh
[R32] 1Ø32:       jmp bildende
[Ø62] 1Ø33: falschmodus:
[1S2] 1Ø34:       lea si,falschmsg
[LK2] 1Ø35:       call writetext
[ENØ] 1Ø36:       stc
[GG1] 1Ø37: bildende:
[G21] 1Ø38:       pop di
[G31] 1Ø39:       pop es
[X31] 1Ø4Ø:       pop dx
[V31] 1Ø41:       pop cx
[T31] 1Ø42:       pop bx
[R31] 1Ø43:       pop ax
[3PØ] 1Ø44:       ret
[D62] 1Ø45: bild          endp
[ØØØ] 1Ø46:
[YY6] 1Ø47: ;Schreibt 5 Anzeigeblöcke und wartet auf
[5M3] 1Ø48: ;Tastendruck [enter]
[KN3] 1Ø49: ausgabe          proc near
[WC1] 1Ø5Ø:       push cx
[F53] 1Ø51:       inc ausgabezaehler
[TD3] 1Ø52:       cmp ausgabezaehler,5
[3B1] 1Ø53:       jne mach
[932] 1Ø54:       lea dx,gibaus
[2A1] 1Ø55:       mov ah,9
[381] 1Ø56:       int 21h
[F43] 1Ø57: endlos:      xor ah,ah
[G81] 1Ø58:       int 16h
[6X1] 1Ø59:       cmp ax,1cØdh
[TS1] 1Ø6Ø:       jne endlos
[3Y6] 1Ø61: ;Funktion Ø von Int 16h wartet auf Tasten-
[YR6] 1Ø62: ;druck, scan-ascii-code des Zeichen wird
[3C6] 1Ø63: ;nach ax übergeben und mit 'enter'(1cØdh)
[XR5] 1Ø64: ;verglichen, kein 'enter' => warten
[7F3] 1Ø65:       mov ausgabezaehler,1
[3E1] 1Ø66:       call bild
[2K2] 1Ø67:       lea si,meldung11
[LK2] 1Ø68:       call writetext
[GU2] 1Ø69: mach:      lea si,alt
[LK2] 1Ø7Ø:       call writetext
[F72] 1Ø71:       mov ax,siflag
[SZ2] 1Ø72:       sub ax,offset tab2
[BA1] 1Ø73:       mov cl,2
[VF1] 1Ø74:       shr ax,cl
[W82] 1Ø75:       mov merker,ax
[792] 1Ø76:       call writehex
[V31] 1Ø77:       pop cx
[HW1] 1Ø78:       mov ax,Øe2Øh
[Y71] 1Ø79:       int 1Øh
[4Z1] 1Ø8Ø:       mov ax,Øe3dh
[Y71] 1Ø81:       int 1Øh
[HW1] 1Ø82:       mov ax,Øe2Øh
[Y71] 1Ø83:       int 1Øh
[672] 1Ø84:       mov si,siflag
[GC1] 1Ø85:       push si
[XS2] 1Ø86:       call writedword
[LM1] 1Ø87:       lea si,neu
[LK2] 1Ø88:       call writetext
[6G1] 1Ø89:       mov si,di
[XS2] 1Ø9Ø:       call writedword
[9U2] 1Ø91:       call neuprogname
[V82] 1Ø92:       mov ax,merker
[LB2] 1Ø93:       cmp merker,28h
[LS1] 1Ø94:       jbe letsgo
[122] 1Ø95:       call except
[VM1] 1Ø96:       jmp umweg
[X23] 1Ø97: letsgo:      push si
[AE1] 1Ø98:       mov si,81
[A31] 1Ø99:       mul si
[XU1] 11ØØ:       lea si,intØ
[ØD1] 11Ø1:       add si,ax
[LK2] 11Ø2:       call writetext
[E31] 11Ø3:       pop si
[HB2] 11Ø4: umweg:      pop si
[672] 11Ø5:       mov si,siflag
[3PØ] 11Ø6:       ret
[VT2] 11Ø7: ausgabe          endp
[ØØØ] 11Ø8:
[4U6] 11Ø9: ;ermittelt aus den MCBs die Namen der TSR-
[FN6] 111Ø: ;Programme,die nach vekdat.dat gestartet
[GD1] 1111: ;wurden
[ZØ4] 1112: neuprogname      proc near
[EC1] 1113:       push es
[3B1] 1114:       push di
[GC1] 1115:       push si
[WC1] 1116:       push cx
[N94] 1117:       mov ax,word ptr ds:[si+2]
[DØ1] 1118:       dec ax
[3H1] 1119:       mov es,ax
[FØ4] 112Ø:       cmp byte ptr es:[Ø],'M'
[KM1] 1121:       je nameaus
[4D1] 1122:       sub ax,16
[3H1] 1123:       mov es,ax
[FØ4] 1124:       cmp byte ptr es:[Ø],'M'
[KM1] 1125:       je nameaus
[WK2] 1126:       lea si,neuprog1
[LK2] 1127:       call writetext
[R12] 1128:       jmp nameend
[ZE1] 1129: nameaus:
[4N3] 113Ø:       cmp word ptr es:[1],Ø
[BK1] 1131:       je mcbfrei
[SC2] 1132:       lea si,neuprog
[LK2] 1133:       call writetext
[8B1] 1134:       mov si,8
[NB1] 1135:       mov cx,8
[VM1] 1136:       mov ah,Øeh
[NS1] 1137: namekreis:
[S74] 1138:       mov al,byte ptr es:[si]
[GV3] 1139:       cmp byte ptr es:[si],Ø
[VK1] 114Ø:       je nameend
[Y71] 1141:       int 1Øh
[911] 1142:       inc si
[FG2] 1143:       loop namekreis
[PC1] 1144: nameend:
[FY1] 1145:       mov ax,Øe Øah
[Y71] 1146:       int 1Øh
[5N1] 1147:       mov al,Ødh
[Y71] 1148:       int 1Øh
[2C1] 1149: mcbfrei:
[V31] 115Ø:       pop cx
[E31] 1151:       pop si
[G21] 1152:       pop di
[G31] 1153:       pop es
[3PØ] 1154:       ret
[V53] 1155: neuprogname      endp
[ØØØ] 1156:
[CØ7] 1157: ;überprüft,ob es sich um einen gültigen
[NY6] 1158: ;Programmnamen handelt,Steuerzeichen und
[8A6] 1159: ;Grafikzeichen werden nicht angezeigt
[5R3] 116Ø: filtern          proc near
[WC1] 1161:       push cx
[GC1] 1162:       push si
[NB1] 1163:       mov cx,8
[8B1] 1164:       mov si,8
[P82] 1165: filtercircle:
[S74] 1166:       mov al,byte ptr es:[si]
[GV3] 1167:       cmp byte ptr es:[si],Ø
[HZ1] 1168:       je filterok
[KK1] 1169:       cmp al,32h
[TP2] 117Ø:       jb nichtanzeigen
[ML1] 1171:       cmp al,7ah
[PP2] 1172:       ja nichtanzeigen
[911] 1173:       inc si
[ZZ2] 1174:       loop filtercircle
[D62] 1175:       jmp filterok
[SE2] 1176: nichtanzeigen:
[XZ1] 1177:       mov ax,9d9dh
[6L1] 1178: filterok:
[E31] 1179:       pop si
[V31] 118Ø:       pop cx
[3PØ] 1181:       ret
[EV2] 1182: filtern          endp
[ØØØ] 1183:
[1Ø7] 1184: ;Übergabe der Offsetadressen von Texten
[RE3] 1185: except          proc near
[DC2] 1186:       cmp merker,2eh
[FUØ] 1187:       ja go1
[NP2] 1188:       lea si,ausnahme1
[LK2] 1189:       call writetext
[3PØ] 119Ø:       ret
[2G3] 1191: go1:      cmp merker,2fh
[N71] 1192:       jne go2
[MP2] 1193:       lea si,ausnahmeØ
[LK2] 1194:       call writetext
[3PØ] 1195:       ret
[CE3] 1196: go2:      cmp merker,32h
[F61] 1197:       ja goin
[NP2] 1198:       lea si,ausnahme1
[LK2] 1199:       call writetext
[3PØ] 12ØØ:       ret
[3E3] 12Ø1: goin:      cmp merker,33h
[1G1] 12Ø2:       jne goin1
```

```
[RT2] 12Ø3:       lea si,ausnahme11
[LK2] 12Ø4:       call writetext
[3PØ] 12Ø5:       ret
[KG3] 12Ø6: goin1:      cmp merker,3fh
[NUØ] 12Ø7:       ja go3
[NP2] 12Ø8:       lea si,ausnahme1
[LK2] 12Ø9:       call writetext
[3PØ] 121Ø:       ret
[CE3] 1211: go3:        cmp merker,4Øh
[X71] 1212:       jne go4
[PP2] 1213:       lea si,ausnahme2
[LK2] 1214:       call writetext
[3PØ] 1215:       ret
[LE3] 1216: go4:        cmp merker,41h
[181] 1217:       jne go5
[RP2] 1218:       lea si,ausnahme3
[LK2] 1219:       call writetext
[3PØ] 122Ø:       ret
[SG3] 1221: go5:        cmp merker,5fh
[YUØ] 1222:       ja go6
[SP2] 1223:       lea si,ausnahme4
[LK2] 1224:       call writetext
[3PØ] 1225:       ret
[EF3] 1226: go6:        cmp merker,66h
[1VØ] 1227:       ja go7
[TP2] 1228:       lea si,ausnahme5
[LK2] 1229:       call writetext
[3PØ] 123Ø:       ret
[NF3] 1231: go7:        cmp merker,67h
[D81] 1232:       jne go8
[NT2] 1233:       lea si,ausnahme1Ø
[LK2] 1234:       call writetext
[3PØ] 1235:       ret
[8H3] 1236: go8:        cmp merker,7fh
[7VØ] 1237:       ja go9
[UP2] 1238:       lea si,ausnahme6
[LK2] 1239:       call writetext
[3PØ] 124Ø:       ret
[VF3] 1241: go9:        cmp merker,85h
[FØ1] 1242:       ja go1Ø
[VP2] 1243:       lea si,ausnahme7
[LK2] 1244:       call writetext
[3PØ] 1245:       ret
[ØG3] 1246: go1Ø:       cmp merker,ØfØh
[LØ1] 1247:       ja go11
[WP2] 1248:       lea si,ausnahme8
[LK2] 1249:       call writetext
[3PØ] 125Ø:       ret
[RG3] 1251: go11:       cmp merker,Øffh
[UC1] 1252:       ja hinaus
[XP2] 1253:       lea si,ausnahme9
[LK2] 1254:       call writetext
[KC2] 1255: hinaus:     ret
[LL2] 1256: except          endp
[ØØØ] 1257:
[GM6] 1258: ;gibt Text mit Interrupt 1Øh,Funktion Øeh
[NS6] 1259: ;aus, Offsetadresse des Text wird in si
[MY1] 126Ø: ;übergeben
[KN3] 1261: writetext       proc near
[PC1] 1262:       push ax
[GC1] 1263:       push si
[VM1] 1264:       mov ah,Øeh
[BK2] 1265: kreisx:       lodsb
[HW1] 1266:       cmp al,Øffh
[381] 1267:       je exit
[Y71] 1268:       int 1Øh
[KV1] 1269:       jmp kreisx
[2A2] 127Ø: exit:       pop si
[R31] 1271:       pop ax
[3PØ] 1272:       ret
[GT2] 1273: writetext       endp
[ØØØ] 1274:
[B35] 1275: ;schreibt Interruptvektor als
[6P3] 1276: ;[Segment]:[Offset]
[VS3] 1277: writedword      proc near
[PC1] 1278:       push ax
[GC1] 1279:       push si
[911] 128Ø:       inc si
[911] 1281:       inc si
[A12] 1282:       inc zaehler
[542] 1283: ofs:        lodsw
[PC1] 1284:       push ax
[NM1] 1285:       xchg ah,al
[792] 1286:       call writehex
[R31] 1287:       pop ax
[792] 1288:       call writehex
[X62] 1289:       cmp zaehler,1
[ØE1] 129Ø:       jne raus
[SY1] 1291:       mov ax,Øe3ah
[Y71] 1292:       int 1Øh
[5Ø2] 1293:       dec zaehler
[K91] 1294:       sub si,4
[RC1] 1295:       jmp ofs
[6A2] 1296: raus:       pop si
[R31] 1297:       pop ax
[3PØ] 1298:       ret
[U13] 1299: writedword      endp
[ØØØ] 13ØØ:
[RK4] 13Ø1: ;wandelt Wert in al nach Hex
[CE3] 13Ø2: writehex        proc near
[PC1] 13Ø3:       push ax
[WC1] 13Ø4:       push cx
[2F1] 13Ø5:       mov ah,al
[DA1] 13Ø6:       mov cl,4
[RE1] 13Ø7:       shr al,cl
[C73] 13Ø8:       call writehexdigit
[6F1] 13Ø9:       mov al,ah
[6K1] 131Ø:       and al,Øfh
[C73] 1311:       call writehexdigit
[V31] 1312:       pop cx
[R31] 1313:       pop ax
[3PØ] 1314:       ret
[NK2] 1315: writehex        endp
[ØØØ] 1316:
[775] 1317: ;schreibt Wert in al als Hexzahl
[PA4] 1318: writehexdigit       proc near
[PC1] 1319:       push ax
[MB1] 132Ø:       cmp al,1Ø
[NA2] 1321:       jae hexletter
[ZF1] 1322:       add al,3Øh
[6K2] 1323:       jmp writedigit
[SV1] 1324: hexletter:
[L91] 1325:       add al,55
[L22] 1326: writedigit:
[YE2] 1327:       call writechar
[R31] 1328:       pop ax
[3PØ] 1329:       ret
[ME3] 133Ø: writehexdigit       endp
[ØØØ] 1331:
[R97] 1332: ;schreibt ein Zeichen,Steuerzeichen werden
[D33] 1333: ;als '.' geschrieben
[GL3] 1334: writechar       proc near
[PC1] 1335:       push ax
[UB1] 1336:       cmp al,32
[VH1] 1337:       jae print
[DG1] 1338:       mov al,'.'
[WY3] 1339: print:  inc charzaehler
[VM1] 134Ø:       mov ah,Øeh
[Y71] 1341:       int 1Øh
[R31] 1342:       pop ax
[3PØ] 1343:       ret
[ER2] 1344: writechar       endp
[ØØØ] 1345:
[KB6] 1346: ;führt carriage return + linefeed aus
[VB3] 1347: carriage       proc near
[PC1] 1348:       push ax
[VY1] 1349:       mov ax,ØeØdh
[Y71] 135Ø:       int 1Øh
[ZM1] 1351:       mov al,Øah
[Y71] 1352:       int 1Øh
[R31] 1353:       pop ax
[3PØ] 1354:       ret
[4G2] 1355: carriage       endp
[ØØØ] 1356:
[9K5] 1357: ;schreibt eine 32-bit Zahl in CX:DX
[2R2] 1358: ;als Dezimalzahl
[M34] 1359: writedecimal       proc near
[PC1] 136Ø:       push ax
[M71] 1361:       or dx,dx
[KA1] 1362:       jz next
[962] 1363:       jmp keinnull
[2C2] 1364: next:    or cx,cx
[XF1] 1365:       jz dezaus
[2L1] 1366: keinnull:
[TC1] 1367:       push bx
[WC1] 1368:       push cx
[ZC1] 1369:       push dx
[GC1] 137Ø:       push si
[DH1] 1371:       mov ax,dx
[EH1] 1372:       mov bx,cx
[1E1] 1373:       mov si,1Ø
[PH1] 1374:       xor cx,cx
[3V2] 1375: kreis1: xor dx,dx
[VR1] 1376:       xchg ax,bx
[W21] 1377:       div si
[VR1] 1378:       xchg ax,bx
[W21] 1379:       div si
[ZC1] 138Ø:       push dx
[P11] 1381:       inc cx
[A71] 1382:       or bx,bx
[MP1] 1383:       jne kreis1
[571] 1384:       or ax,ax
[MP1] 1385:       jne kreis1
[X72] 1386: ring:       pop ax
[C73] 1387:       call writehexdigit
[7K1] 1388:       loop ring
[E31] 1389:       pop si
[X31] 139Ø:       pop dx
[V31] 1391:       pop cx
[T31] 1392:       pop bx
[ØF1] 1393:       jmp goon
[Z93] 1394: dezaus: mov ax,Øe3Øh
[Y71] 1395:       int 1Øh
[G72] 1396: goon:    pop ax
[3PØ] 1397:       ret
[X73] 1398: writedecimal       endp
[ØØØ] 1399:
[B62] 14ØØ: code            ends
[4W1] 14Ø1: end      start
```

Listing. »vekdat.asm« merkt sich alle Interrupt-Vektoren, um sie nach Aktivierung von speicherresidenten Programmen wiederherstellen zu können

Turbo Pascal ab Version 5.0

Kopieren in einem Zug

Beim Kopieren von Disketten mit den MS-DOS-Programmen »xcopy« oder »diskcopy« ist der häufig notwendige Diskettenwechsel sehr lästig. Der Grund dafür liegt darin, daß der Inhalt einer Diskette bei den großen Formaten nicht komplett in den Arbeitsspeicher paßt. Warum benutzt man deshalb nicht die Festplatte als Hilfsspeicher?

Der Vorteil dafür liegt klar auf der Hand. Man müßte jede Diskette nur einmal einlegen, da auf der Festplatte auch 1,44 MByte auf einmal Platz haben. Genau das kann das Programm »fpcopy.pas« (Listing). Es kopiert eine Diskette komplett auf die Festplatte, einschließlich der FAT und des Boot-Blocks. Danach können beliebig viele Kopien von der Originaldiskette erzeugt werden.
»fpcopy« unterstützt alle MS-DOS-Formate von 180 KByte bis 1,44 MByte mit Ausnahme des alten und wenig verbreiteten 320-KByte-Formates.
Der Aufruf des Programms lautet:

```
fpcopy a: c:\temp
```

Das Programm interpretiert den ersten Parameter als das Laufwerk, in dem sich die zu kopierende Diskette befindet. Der zweite Parameter ist der Name der temporären Datei, in der die Diskette abgelegt wird. In diesem Beispiel wird die Diskette in Laufwerk A eingelesen und, vorausgesetzt es ist ausreichend Platz vorhanden, in die Datei »c:\temp« kopiert. Anschließend erfolgt eine Aufforderung zum Diskettenwechsel. Sollte sich bereits eine Datei mit gleichem Namen auf der Festplatte befinden, werden Sie darauf aufmerksam gemacht und können abbrechen. Ist die neu eingelegte Diskette in Laufwerk A vom gleichen Format wie das Original, wird die Kopie erzeugt, im anderen Fall bricht »fpcopy« mit einer Fehlermeldung ab. Danach können Sie weitere Kopien anfertigen. Hat man die gewünschte Anzahl Kopien erzeugt, wird die Datei »c:\temp« automatisch wieder gelöscht.
Der Kopiervorgang läßt sich durch einen einfachen Trick noch zusätzlich beschleunigen. Dazu verwenden Sie anstelle der Festplatte eine RAM-Disk als Zwischenspeicher. Diese muß selbstverständlich ausreichend groß dimensioniert sein. Beachten Sie jedoch, daß die Daten nach einem Reset oder Ausschalten des Computers verloren sind. Der sichere Weg ist also die Festplatte!
Man kann »fpcopy« jedoch nicht nur ausschließlich als Kopierprogramm verwenden. Da die Diskette in einer Datei gespeichert wird, lassen sich somit auch komplette Disketten dauerhaft auf der Festplatte ablegen. Wenn Sie dies wollen, verwenden Sie im Aufruf den Parameter »/keep«. Dadurch wird nach dem Kopieren der Diskette in Laufwerk A auf die Festplatte in die Datei »c:\temp« abgebrochen. Für unser Beispiel geben Sie dann ein:

```
fpcopy a: c:\temp /keep
```

Hierzu noch ein Tip für Freunde der Datenfernübertragung. Wenn Sie die durch »fpcopy« erzeugte Datei mit einem der im Sharewarevertrieb erhältlichen Komprimierungsprogramme wie »pkzip« oder »lharc« in ihrem Umfang verkleinern, können Sie die so entstandene Datei jederzeit per Modem verschicken. Ihre Gegenstation braucht dann selbstverständlich auch »fpcopy«, um die Datei wieder auf Diskette kopieren zu können.
Wenn Sie eine von »fpcopy« generierte Datei wieder auf Diskette kopieren wollen, geben Sie »/use« an.
Beispiel:

```
fpcopy a: c:\temp /use
```

Dadurch wird die Datei »c:\temp« wieder auf die Diskette in Laufwerk A geschrieben. Sie wird danach nicht gelöscht und bleibt somit für weitere Kopien erhalten. Wenn sie nicht mehr benötigt wird, kann sie mit dem MS-DOS-Befehl »del« ganz normal gelöscht werden.

(Kai Engert/hs)

```
[KV3]    1: { Programm: fpcopy.pas
[WB6]    2:   Funktion: vervielfältigt Disketten
[YY5]    3:   Sprache:  Turbo Pascal ab Version 5.0
[C33]    4:   Autor:    Kai Engert
[P35]    5:   (c)1991 DMV Widuch GmbH & Co. KG }
[000]    6:
[5D3]    7:  {$m 16384,61440,655360}
[000]    8:
[S62]    9: USES Dos,Crt;
[000]   10:
[ST0]   11: TYPE
[T06]   12:  puffertyp = ARRAY [0..61439] OF BYTE;
[000]   13:
[V06]   14: VAR sek             : ARRAY[0..511] OF BYTE;
[YE2]   15:     genugspeicher,
[XF1]   16:     keepmode,
[WF3]   17:     usemode         : BOOLEAN;
[Y82]   18:     bytesprosek,
[VE1]   19:     zielbps,
[EZ1]   20:     sekaufdisk,
[YK2]   21:     zielsad,bedarf,
[0A2]   22:     platzauftemp,
[5K3]   23:     heapfrei        : LONGINT;
[7S2]   24:     f,init          : FILE;
[EU2]   25:     i               : INTEGER;
[ZG3]   26:     tempfile        : STRING;
[4V4]   27:     heapzeiger      : ARRAY[0..29] OF
[B62]   28:                       ^puffertyp;
[VA2]   29:     anzahlzeiger,
[B72]   30:     templaufnum,
[R33]   31:     laufnum         : BYTE;
[SA2]   32:     templaufwerk,
[ST4]   33:     backuplaufwerk : STRING[1];
[UM3]   34:     templauf,lauf,c: CHAR;
[000]   35:
[L23]   36: PROCEDURE fanfare;
[6E2]   37: VAR q:INTEGER;
[MU0]   38: BEGIN
[5C3]   39:   FOR q:= 100 TO 10000 DO
[SG1]   40:     Sound(q);
[FN3]   41:   FOR q:= 10000 DOWNTO 0 DO
[SG1]   42:     Sound(q);
[3G1]   43:   NoSound;
[ZN0]   44: END;
[000]   45:
[XC5]   46: PROCEDURE stop(ausgabe:STRING);
[MU0]   47: BEGIN
[YD3]   48:   WriteLn(#10,ausgabe);
[DU0]   49:   Halt;
[ZN0]   50: END;
[000]   51:
[4C6]   52: PROCEDURE tastdruck(ausgabe:STRING);
[MU0]   53: BEGIN
[ZD3]   54:   WHILE KeyPressed DO
[702]   55:     c:= ReadKey;
[YD3]   56:   WriteLn(#10,ausgabe);
[944]   57:   WHILE NOT KeyPressed DO;
```

Listing. »fpcopy.pas« löst auch Ihre Kopierprobleme

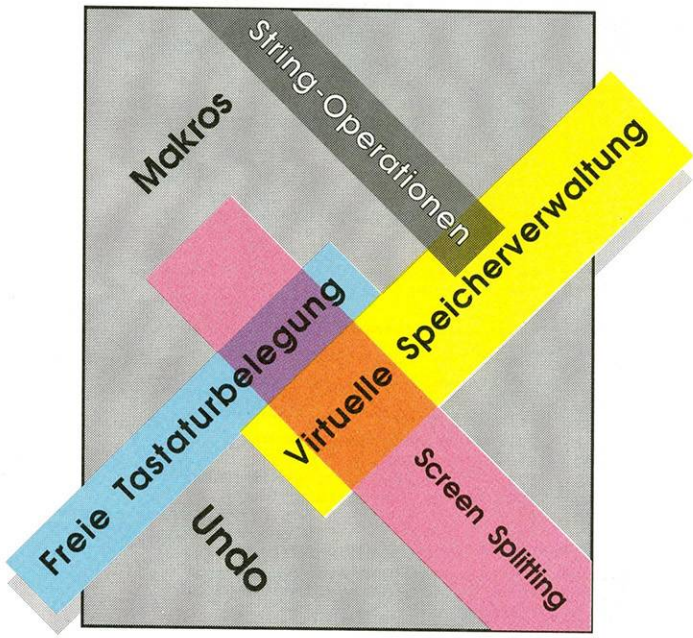
```
[ZNØ]   58: END;
[ØØØ]   59:
[X86]   6Ø: PROCEDURE lies(laufnum,seknum,anzahl:
[V74]   61:                INTEGER;zeiger:POINTER);
[UM2]   62: VAR err : WORD;
[ØØØ]   63:
[MUØ]   64: BEGIN
[575]   65:   INLINE($1E/$C5/$5E/<zeiger/$8B/$4E);
[VØ6]   66:   INLINE(<anzahl/$8B/$56/<seknum/$8A/$46);
[9U5]   67:   INLINE(<laufnum/$55/$CD/$25/$72/$Ø2/$33);
[M65]   68:   INLINE($CØ/$9D/$5D/$89/$46/<err/$1F);
[BX3]   69:   IF (err DIV 256) <> Ø THEN
[ZA4]   7Ø:     stop('Fehler beim Lesen !');
[ZNØ]   71: END;
[ØØØ]   72:
[SU2]   73: PROCEDURE lesen;
[BX5]   74: VAR sek1,sek2,maxsek,Pos : INTEGER;
[HP3]   75:     ende                 : BOOLEAN;
[ØØØ]   76:
[MUØ]   77: BEGIN
[884]   78:   maxsek:= anzahlzeiger*12Ø;
[871]   79:   Pos:= Ø;
[AX2]   8Ø:   sek1:= sekaufdisk;
[712]   81:   ende:= FALSE;
[156]   82:   Write(#1Ø,'   % des Disketten-Inhalt''s',
[ZH5]   83:         ' wurden bis jetzt eingelesen...');
[YY2]   84:   GotoXY(1,WhereY);
[P44]   85:   IF NOT genugspeicher THEN
[7L2]   86:     Rewrite(f,6144Ø);
[P31]   87:   REPEAT
[ST2]   88:     Dec(sek1,maxsek);
[T82]   89:     IF sek1>=Ø THEN
[CF3]   9Ø:       sek2:= anzahlzeiger-1
[ZRØ]   91:     ELSE
[MUØ]   92:       BEGIN
[9R4]   93:         sek2:= ((sek1+maxsek)DIV 12Ø)-1;
[UX1]   94:         ende:= TRUE;
[ZNØ]   95:       END;
[F23]   96:     FOR i:= Ø TO sek2 DO
[MUØ]   97:       BEGIN
[FW5]   98:         Write(Trunc(Pos*1ØØ.Ø/sekaufdisk));
[YY2]   99:         GotoXY(1,WhereY);
[GC3]  1ØØ:         lies(laufnum,Pos,12Ø,
[RR2]  1Ø1:              heapzeiger[i]);
[CØ2]  1Ø2:         Inc(Pos,12Ø);
[ZNØ]  1Ø3:       END;
[P44]  1Ø4:     IF NOT genugspeicher THEN
[F23]  1Ø5:       FOR i:= Ø TO sek2 DO
[PG5]  1Ø6:         BlockWrite(f,heapzeiger[i]^,1);
[AY1]  1Ø7:   UNTIL ende;
[P44]  1Ø8:   IF NOT genugspeicher THEN
[KD1]  1Ø9:     Close(f);
[5B1]  11Ø:   ClrEoL;
[ZNØ]  111: END;
[ØØØ]  112:
[417]  113: PROCEDURE schreib(laufnum,seknum,anzahl:
[V74]  114:                   INTEGER;zeiger:POINTER);
[UM2]  115: VAR err : WORD;
[MUØ]  116: BEGIN
[575]  117:   INLINE($1E/$C5/$5E/<zeiger/$8B/$4E);
[VØ6]  118:   INLINE(<anzahl/$8B/$56/<seknum/$8A/$46);
[DU5]  119:   INLINE(<laufnum/$55/$CD/$26/$72/$Ø2/$33);
[M65]  12Ø:   INLINE($CØ/$9D/$5D/$89/$46/<err/$1F);
[BX3]  121:   IF (err DIV 256) <> Ø THEN
[U15]  122:     stop('Fehler beim Schreiben !');
[ZNØ]  123: END;
[ØØØ]  124:
[LH3]  125: PROCEDURE schreiben;
[BX5]  126: VAR sek1,sek2,maxsek,Pos : INTEGER;
[HP3]  127:     ende                 : BOOLEAN;
[ØØØ]  128:
[MUØ]  129: BEGIN
[884]  13Ø:   maxsek:= anzahlzeiger*12Ø;
[871]  131:   Pos:= Ø;
[AX2]  132:   sek1:= sekaufdisk;
[712]  133:   ende:= FALSE;
[P44]  134:   IF NOT genugspeicher THEN
[K92]  135:     Reset(f,6144Ø);
[WH5]  136:   Write(#1Ø,'   % der Daten wurden bis',
[5Ø6]  137:         ' jetzt auf die Diskette kopiert');
[YY2]  138:   GotoXY(1,WhereY);
[P31]  139:   REPEAT
[ST2]  14Ø:     Dec(sek1,maxsek);
[T82]  141:     IF sek1>=Ø THEN
[CF3]  142:       sek2:= anzahlzeiger-1
[ZRØ]  143:     ELSE
[MUØ]  144:       BEGIN
[HR4]  145:         sek2:= ((sek1+maxsek) DIV 12Ø)-1;
[UX1]  146:         ende:= TRUE;
[ZNØ]  147:       END;
[P44]  148:     IF NOT genugspeicher THEN
[F23]  149:     FOR i:= Ø TO sek2 DO
[U45]  15Ø:       BlockRead(f,heapzeiger[i]^,1);
[F23]  151:     FOR i:= Ø TO sek2 DO
```

Listing. »fpcopy.pas« löst auch Ihre Kopierprobleme

Demnächst bei DMV

Prof Ed

Der Editor für Programmierer



Ob Sie programmieren, einfache Texte schreiben oder Ihre Config.sys- oder Autoexec.bat- Datei umändern wollen:
ProfEd ist das ideale Werkzeug für Sie.

Und das bietet ProfEd:

- Bis zu fünf Dateien lassen sich gleichzeitig bearbeiten.
- Mehrere MByte lange Dateien sind kein Problem für ProfEd.
- Jede Taste der Tastatur kann mit einem oder mehreren Kommandos versehen werden.
- Makros gestatten beliebig viele Editorkommandos abzuarbeiten.
- Mehrere Undo-Funktionen lassen den letzten Arbeitsschritt rückgängig machen.
- Teile einer oder verschiedener Dateien lassen sich gleichzeitig am Bildschirm zur Bearbeitung bereitstellen.
- Suchen und Ersetzen von Zeichenfolgen ist überhaupt kein Problem.
- Zeilen- und spaltenweises Löschen, Kopieren, Verschieben und Füllen von markierten Textbereichen gestatten ein schnelles Bearbeiten Ihrer Dateien. Diese Funktion arbeitet dateiübergreifend und ermöglicht jede nur denkbare Textmanipulation.
- Eine spezielle Netzwerkversion ist erhältlich.

Systemvoraussetzung:
PC, XT, AT, 386er oder 486er Rechner mit mindestens 256 KByte RAM, DOS ab Version 3.0

ProfEd DM **99,–***

Demnächst bei DMV:
Fontline DM **299,–***
Softfontmanipulation und Druckertreibergenerierung

* Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland DM 4,– bzw. für das Ausland DM 6,– Porto und Verpackung.
Bitte benutzen Sie die Bestellkarte.

DMV-Verlag • Postfach 250 • 3440 Eschwege

```
[MUØ]  152:        BEGIN
[FW5]  153:          Write(Trunc(Pos*1ØØ.Ø/sekaufdisk));
[YY2]  154:          GotoXY(1,WhereY);
[F24]  155:          schreib(laufnum,Pos,12Ø,
[RR2]  156:                  heapzeiger[i]);
[CØ2]  157:          Inc(Pos,12Ø);
[ZNØ]  158:        END;
[AY1]  159:    UNTIL ende;
[P44]  16Ø:    IF NOT genugspeicher THEN
[KD1]  161:      Close(f);
[5B1]  162:    ClrEoL;
[ZNØ]  163: END;
[ØØØ]  164:
[CB4]  165: FUNCTION gibtstemp:WORD;
[ØØØ]  166:
[MUØ]  167: BEGIN
[DXØ]  168:   {$i-}
[HT1]  169:   Reset(f,1);
[BXØ]  17Ø:   {$i+}
[NP3]  171:   gibtstemp:= IOResult;
[ZNØ]  172: END;
[ØØØ]  173:
[NV2]  174: PROCEDURE check1;
[ØØØ]  175:
[MUØ]  176: BEGIN
[LF2]  177:   IF usemode THEN
[MUØ]  178:     BEGIN
[N83]  179:       IF gibtstemp<>Ø THEN
[LK2]  18Ø:         stop(tempfile +
[MY3]  181:              ' ist nicht lesbar oder'+
[HC3]  182:              ' nicht vorhanden !');
[VB3]  183:       bedarf:= FileSize(f);
[KD1]  184:       Close(f);
[9T3]  185:       genugspeicher:= FALSE;
[4MØ]  186:     END
[ZRØ]  187:   ELSE
[MUØ]  188:     BEGIN
[P43]  189:       IF gibtstemp=Ø THEN
[MUØ]  19Ø:         BEGIN
[MC1]  191:           fanfare;
[CE4]  192:           tastdruck('Beachten Sie : '+
[YK3]  193:                     tempfile+' existiert'+
[ZV2]  194:                     ' bereits !!!'+#1Ø+#13+
[PT2]  195:                     'Beliebige Taste'+
[2K3]  196:                     ' drücken um fortzu'+
[X72]  197:                     'fahren'+#1Ø+#13+
[222]  198:                     'oder CTRL-C'+
[A72]  199:                     ' drücken um'+
[S83]  2ØØ:                     ' abzubrechen...'+#1Ø);
[ZNØ]  2Ø1:         END;
[DXØ]  2Ø2:     {$i-}
[3Ø2]  2Ø3:     Reset(init);
[UY1]  2Ø4:     Close(init);
[BXØ]  2Ø5:     {$i+}
[Z72]  2Ø6:     i:= IOResult;
[DR3]  2Ø7:     lies(laufnum,Ø,1,@sek);
[685]  2Ø8:     sekaufdisk:= sek[19]+256*sek[2Ø];
[ER5]  2Ø9:     bytesprosek:= sek[11]+256*sek[12];
[2P5]  21Ø:     bedarf:= sekaufdisk*bytesprosek;
[P93]  211:     IF (NOT keepmode) AND
[SU3]  212:       (bedarf<=heapfrei) THEN
[MUØ]  213:       BEGIN
[9P3]  214:         genugspeicher:= TRUE;
[GH4]  215:         WriteLn('Der speicher ist',
[274]  216:                 ' groß genug, ',tempfile,
[EY3]  217:                 ' wird nicht benötigt.');
[4MØ]  218:       END
[ZRØ]  219:     ELSE
[MUØ]  22Ø:       BEGIN
[9T3]  221:         genugspeicher:= FALSE;
[V54]  222:         templaufwerk:= tempfile;
[VS1]  223:         templauf:=
[KD5]  224:            CHAR(Hi(INTEGER(templaufwerk)));
[LB2]  225:         templaufnum:=
[P14]  226:                  Ord(UpCase(templauf))-64;
[LG2]  227:         platzauftemp:=
[TV3]  228:                    DiskFree(templaufnum);
[HP4]  229:         IF bedarf>platzauftemp THEN
[ZN4]  23Ø:           stop('Zuwenig Platz für '+
[UP1]  231:                tempfile);
[ZNØ]  232:       END;
[ZNØ]  233:     END;
[766]  234:   WriteLn(#1Ø,'Kapazität des Backup''s: ',
[G64]  235:          bedarf DIV 1Ø24,' Kilobyte');
[ZNØ]  236: END;
[ØØØ]  237:
[354]  238: FUNCTION check2:BOOLEAN;
[ØØØ]  239:
[MUØ]  24Ø: BEGIN
[DXØ]  241:   {$i-}
[3Ø2]  242:   Reset(init);
[UY1]  243:   Close(init);
[BXØ]  244:   {$i+}
[Z72]  245:   i:= IOResult;
[9E2]  246:   check2:= FALSE;
[DR3]  247:   lies(laufnum,Ø,1,@sek);
[EV4]  248:   zielsad:= sek[19]+256*sek[2Ø];
[BX4]  249:   zielbps:= sek[11]+256*sek[12];
[T65]  25Ø:   IF zielsad*zielbps <> bedarf THEN
[D46]  251:     WriteLn(#1Ø,'Auf diese Diskette kann',
[UU4]  252:             ' nicht kopiert werden, sie',
[EK4]  253:             ' hat ein anderes Format als',
[HØ5]  254:             ' dieQuelldiskette, benötigt',
[HA4]  255:             ' wird eine Diskette mit ',
[U24]  256:             bedarf DIV 1Ø24,' Kilobyte')
[ZRØ]  257:   ELSE
[U82]  258:     check2:= TRUE;
[ZNØ]  259: END;
[ØØØ]  26Ø:
[653]  261: PROCEDURE getmode;
[ØØØ]  262:
[MUØ]  263: BEGIN
[VE6]  264:   WriteLn(#1Ø,#1Ø,'Willkommen bei FPCOPY',
[262]  265:          ' v2.Ø !',#1Ø,#13);
[SR2]  266:   IF (ParamCount<2)
[HW2]  267:     OR  (ParamCount>3)
[V13]  268:     OR  ((ParamCount=3)
[TV3]  269:     AND (ParamStr(3)<>'/use')
[1W4]  27Ø:     AND (ParamStr(3)<>'/keep')) THEN
[UØ6]  271:     stop ('Man startet es durch -> FPCOPY'+
[153]  272:           ' <drive:> <fpfile>'+
[US2]  273:           ' [/keep|/use]');
[FP2]  274:   usemode:= FALSE;
[3S2]  275:   keepmode:= FALSE;
[Z93]  276:   IF ParamCount=3 THEN
[L54]  277:     IF ParamStr(3)='/use' THEN
[HD2]  278:       usemode:= TRUE
[ZRØ]  279:   ELSE
[PM2]  28Ø:     keepmode:= TRUE;
[YV4]  281:   backuplaufwerk:= ParamStr(1);
[C65]  282:   tempfile:= FExpand(ParamStr(2));
[P93]  283:   heapfrei:= MaxAvail;
[FL5]  284:   anzahlzeiger:= heapfrei DIV 6144Ø;
[CP3]  285:   IF anzahlzeiger=Ø THEN
[UR6]  286:   stop('Zu wenig freier Speicherplatz...');
[5Z4]  287:   FOR i:= Ø TO (anzahlzeiger-1) DO
[6F3]  288:     New(heapzeiger[i]);
[VP4]  289:   lauf:= UpCase(CHAR(Hi(INTEGER(
[H63]  29Ø:                          backuplaufwerk))));
[VS3]  291:   laufnum:= Ord(lauf)-65;
[GL5]  292:   Write('Es erfolgt ein Backup von ');
[LF2]  293:   IF usemode THEN
[4F6]  294:     WriteLn(tempfile,' zum Laufwerk ',lauf)
[ZRØ]  295:   ELSE
[R85]  296:     WriteLn('Laufwerk ',lauf,' nach ',
[UP1]  297:             tempfile);
[T45]  298:   IF NOT(keepmode OR usemode) THEN
[176]  299:     WriteLn('und (beliebig oft) zurück.');
[ZNØ]  3ØØ: END;
[ØØØ]  3Ø1:
[MUØ]  3Ø2: BEGIN
[LD1]  3Ø3:   getmode;
[UG5]  3Ø4:   tastdruck('Bitte eine beliebige '+
[F53]  3Ø5:             'Taste drücken...');
[P73]  3Ø6:   Assign(f,tempfile);
[5K5]  3Ø7:   Assign(init,backuplaufwerk+':\');
[A61]  3Ø8:   check1;
[V23]  3Ø9:   IF NOT usemode THEN
[MUØ]  31Ø:     BEGIN
[521]  311:       lesen;
[MC1]  312:       fanfare;
[ZNØ]  313:     END;
[X73]  314:   IF NOT keepmode THEN
[P31]  315:     REPEAT
[225]  316:       tastdruck('Bitte legen Sie die'+
[KB5]  317:       ' Zieldiskette ein, danach Taste'+
[Z92]  318:       ' drücken...');
[VW3]  319:       IF check2 THEN schreiben;
[ZD3]  32Ø:         WHILE KeyPressed DO
[7Ø2]  321:           c:= ReadKey;
[MC1]  322:       fanfare;
[MN5]  323:       WriteLn('noch eine Kopie ??? (j/n)');
[P31]  324:       REPEAT
[7Ø2]  325:         c:= ReadKey;
[R14]  326:       UNTIL c IN ['j','J','n','N'];
[AG3]  327:     UNTIL (c='n') OR (c='N');
[4H6]  328:   IF (NOT genugspeicher)AND(NOT keepmode)
[GP3]  329:     AND (NOT usemode) THEN
[MUØ]  33Ø:     BEGIN
[YE3]  331:       WriteLn(#1Ø,tempfile,
[N84]  332:               ' wird wieder gelöscht.');
[GC1]  333:       Erase(f);
[ZNØ]  334:     END;
[CU4]  335:   WriteLn('Ende von FPCOPY...');
[KNØ]  336: END.
```

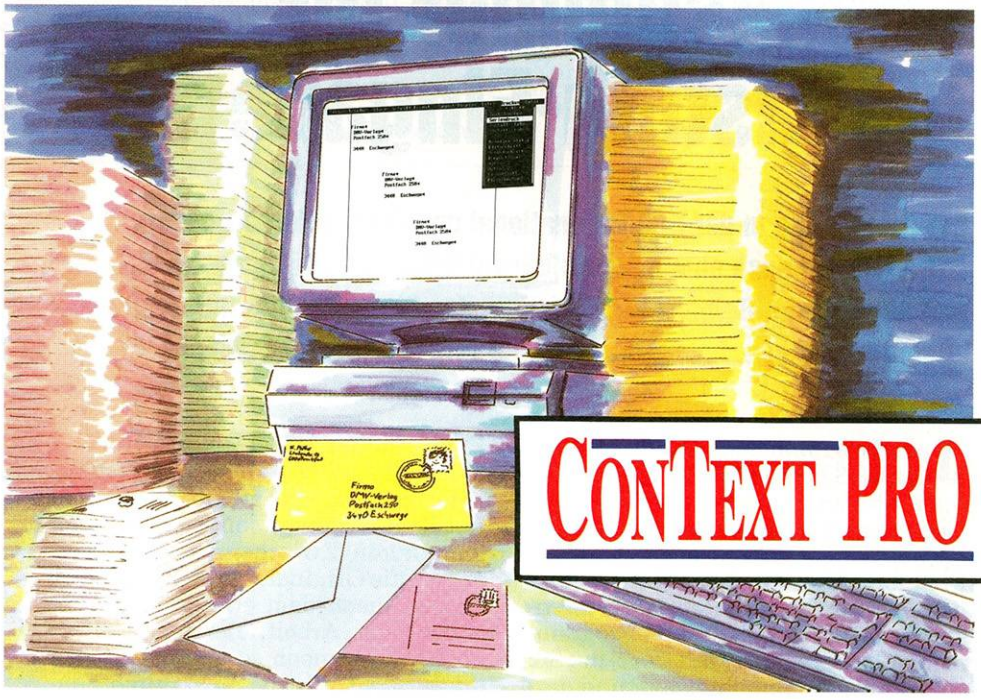
Listing. »fpcopy.pas« löst auch Ihre Kopierprobleme

Textverarbeitung mit Konzept

ConText PRO

Textverarbeitungsprogramm

Leistungsfähige, durchdachte Textverarbeitung mit breit angelegtem Funktionsspektrum und anwendergerechter Handhabung – das ist das Erfolgskonzept von **ConText PRO**.



Komfortable Texterstellung für den universellen Einsatz mit umfangreichen Standardfunktionen:

Vielfältige Schriftgestaltung durch Unterstreichen, Fettschrift, Breitschrift, Kursivschrift, hoch- und tiefgestellte Schrift in WYSIWYG-Darstellung • Blockgrafik-Editor mit Rechenfunktion für Statistiken und Tabellen • Zeilenlineal, automatischer Zeilenumbruch • Blocksatz, Flattersatz und Zeilenzentrierung • Suchen und Ersetzen, Suchbegriffe mit Sonderzeichen oder Joker • Zweiseitiges Layout • Kopf- und Fußzeilen • Unterschiedliche Löschfunktionen • Spalten- und Zeilenblockbefehle • Formatieren: Absätze und fortlaufenden Text • Makro-Funktion • Undo-Funktion • Veränderliche Bildschirmanzeige mit bis zu 60 Zeilen • Zwischenablage für Textabschnitte • Mehrfachausdrucke, Drucken im Hintergrund • Hilfe-Fenster bei allen Funktionen • Serienbrieffunktion • und vieles mehr .

DM 199,–*

Leistungsfähige Zusatzprogramme für ConText PRO und ConText EASY:

ConText-Rechtschreibhilfe

Eine gute und praktische Idee: Während Ihre Hände und Augen mit der Texteingabe beschäftigt sind, warnt Sie die **ConText-Rechtschreibhilfe** durch akustische Signale. Sofort und ohne Rücklauf können Sie gleich an der Cursorposition korrigieren. Lexika können editiert und individuell für Ihre Anwendungen erstellt werden.

DM 69.–*

ConText-Zeicheneditor

Sprengen Sie den ASCII-Zeichensatz: Mit dem **ConText-Zeicheneditor** können Sie eigene Zeichen zur WYSIWYG-Bildschirmdarstellung und für Ihren Drucker für **ConText PRO** und **ConText EASY** erzeugen. Fremdsprachliche oder wissenschaftliche Sonderzeichen sind von nun an kein Problem mehr.

DM 99.–*

ConText-Adreßverwaltung

Adressen praxisgerecht archivieren, sortieren, auflisten, suchen und per Tastendruck in Ihren Text übernehmen – das ist **ConText-Adreßverwaltung**. In Zusammenarbeit mit **ConText PRO** und **ConText EASY** können Ihre Adressen auch direkt für Serienbriefe verarbeitet werden.

DM 49.–*

* Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland DM 4.- bzw. für das Ausland DM 6.- Porto und Verpackung.
Bitte benutzen Sie die Bestellkarte.

DMV-Verlag • Postfach 250 • 3440 Eschwege

DMV Daten- und Medienverlag

SCheck 2.02

Checksummer für die Listing-Eingabe

Die Programme in der DOS International und der Sonderheftreihe DOS Extra werden nach einem speziellen Verfahren mit Prüfsummen versehen und abgedruckt. Beinahe jedes Listing, das Sie in der DOS International abgedruckt sehen, beinhaltet auch die Prüfsummen für SCheck.

Jede Programmzeile in den Listings der DOS beginnt mit einer dreistelligen Prüfsumme. Sie ist in eckige Klammern gesetzt und wird beim Abtippen nicht mit eingegeben. Nach der Klammer folgt ein Leerzeichen, und darauf der eigentliche Text der Programmzeile (nur bei Basic-Programmen) oder eine Zeilennummer, die Sie ebenfalls nicht mit eingeben, und darauf erst der Programmtext. Nur diesen letzten Teil der abgedruckten Zeile geben Sie beim Abtippen ein.

Bei allen Programmzeilen wird im Abdruck nach jeweils 55 Zeichen ein Zeilenumbruch vorgenommen. Die umgebrochene Teilzeile erhält dabei keine Prüfsumme. Bei der Eingabe geben Sie umgebrochene Zeilen in einem Stück ein, drücken also nicht die Enter-Taste wenn ein Zeilenumbruch im Ausdruck steht, sondern nur, wenn eine neue Zeilennummer auftaucht.

Beim Abtippen von Programmen sind Tippfehler kaum zu vermeiden. Mit dem Checksummenprogramm »SCheck« überprüfen Sie die abgetippten Listings auf eventuelle Fehler. Das Programm erlaubt nicht nur die Diagnose von fehlerhaft eingegebenen Zeilen, sondern mittels eines integrierten Editors auch gleich die Korrektur der fehlerhaften Zeilen.

SCheck wird unter GW-Basic mit:

```
run "scheck"
```

gestartet. Als erstes wählen Sie die gewünschte Prüfsummenversion. In der vorliegenden DOS-Ausgabe brauchen Sie immer »SCheck Version 2.0«. Nur wer Programme aus Ausgaben der DOS International vor Oktober 1988 eingeben will, muß den ebenfalls integrierten Prüfsummenalgorithmus »SCheck Version 1.0« aufrufen. Danach geben Sie den gewünschten Prüfmodus – normalerweise »Standard« – an und abschließend den Programmnamen des Listings, so wie Sie es auf dem Datenträger abgelegt haben.

Zuvor sollten Sie sich aber sicherheitshalber eine Kopie Ihres Listings anfertigen. SCheck arbeitet zwar nicht mit dem Original, sondern mit einer temporären Zwischendatei und verändert die Originaldatei erst ganz zum Schluß, nach dem erfolgreichen Abschluß der Arbeit. Dennoch sollten Sie auch ein noch so geringes Risiko eines Datenverlusts, etwa durch Fehler beim Speichern, ausschalten.

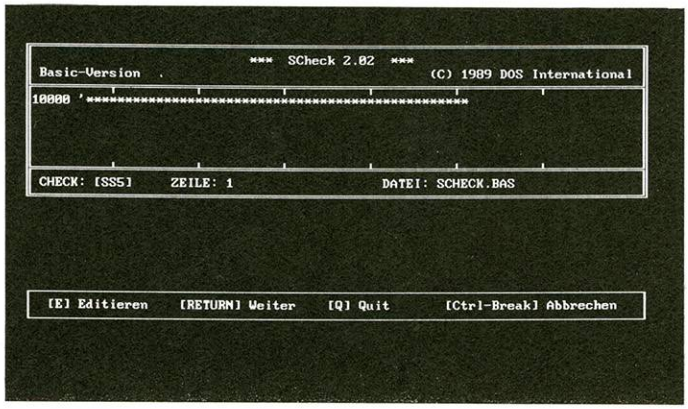
Wie Sie im Bild sehen, zeigt SCheck die aktuelle Zeile stets in einem besonderen Bildschirmfenster an. Die Prüfsumme erscheint links unter dem Fenster. Beim Überprüfen Ihres Listings vergleichen Sie die angezeigte Prüfsumme mit der in der DOS-Ausgabe in eckigen Klammern abgedruckten Prüfsumme. Rechts daneben finden Sie die aktuelle Zeilennummer und den Dateinamen. Im unteren Fenster werden Fehlermeldungen und Hinweise auf die weitere Bearbeitung ausgegeben. Im abgebildeten Fall können Sie mit [Enter] die Zeile übernehmen oder mit [E] den Editor aufrufen.

Der Editor dient der Korrektur fehlerhafter Zeilen. Er arbeitet immer im Einfügemodus, so daß fehlerhafte Eingaben extra gelöscht werden müssen. Mit den Cursor-Tasten steuern Sie den Cursor im Editierfenster. [Home] bringt den Cursor an den Anfang der angezeigten Zeile, [End] entsprechend ans Ende. Löschen können Sie das Zeichen links vom Cursor mit der großen Löschtaste oberhalb der Enter-Taste, [Del] entfernt das Zeichen unter dem Cursor. Mit [Enter] übernehmen Sie die korrigierte Zeile. Steuerzeichen wie zum Beispiel Tabulatoren dürfen in einer Zeile nicht verwendet werden. Haben Sie so das gesamte Programm oder auch nur »verdächtige« Teile davon durchgearbeitet, so speichert ein letzter Druck auf die Enter-Taste das korrigierte Listing. SCheck wird beendet und das fehlerfreie Programmlisting steht zu Ihrer Verfügung.

SCheck erlaubt aber noch einen zweiten Weg, das Listing korrekt einzutippen. Dabei geben Sie die Prüfsumme gleich bei der Eingabe ein. Die eckigen Klammern lassen Sie aber weg. Hinter der dreistelligen Prüfsumme fügen Sie ein Leerzeichen ein und danach den normalen Programmtext dieser Zeile. Wenn Sie den Text so aufbereitet vorliegen haben, können Sie die automatische Überprüfung mit SCheck nutzen. Rufen Sie dazu die Option »Mit Prüfsumme« auf.

SCheck verwandelt das Listing nun in eine Quelldatei für den jeweiligen Compiler oder Interpreter, indem die drei Prüfsummenstellen und das Leerzeichen vom Rest der Zeile abgetrennt werden. Aus der Programmzeile berechnet SCheck die Prüfsumme und vergleicht sie mit der zuvor abgetrennten Summe, die Sie eingegeben haben. Falls ein Unterschied auftritt, haben Sie Gelegenheit, die fehlerhafte Zeile manuell zu korrigieren.

Nach einem solchen Durchlauf von SCheck steht wieder das fehlerfreie Programm, diesmal allerdings ohne Prüfsumme, auf der Diskette. Auch hier gilt, daß Sie nicht mit dem Original arbeiten sollten, sondern nur mit einer Kopie, um im Zweifelsfall nochmal von vorne beginnen zu können. SCheck verfügt aber noch über zwei weitere Optionen. Die von Fall zu Fall abgedruckten hexadezimalen Listings können Sie nämlich mit SCheck im Binärmodus direkt in lauffähige Programme umwandeln. Dazu müssen Sie das Hexdump-Listing mit einem



Der Arbeitsbildschirm von SCheck in Aktion

ganz normalen Editor (zum Beispiel Turbo Pascal, Wordstar oder Edlin) als Textdatei eingeben. Anschließend überprüfen Sie mit SCheck zunächst die Korrektheit der abgetippten Daten wie bei jedem anderen Listing. Befindet sich das Programm korrekt auf Diskette, so wird die Option »Hexdump« aufgerufen. SCheck wird nun dazu veranlaßt, eine Hexdump-Textdatei in ein ablauffähiges Programm umzuwandeln. SCheck erzeugt in jedem Fall eine Datei mit der Namenserweiterung »bin«. Um daraus eine lauffähige Datei zu machen, müssen Sie sie mit dem Ren-Befehl von MS-DOS umbenennen:

```
ren <name>.bin <name>.com
```

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

```
ren <name>.bin <name>.exe
```

Verwenden Sie nach dem Abtippen des Listings aus der DOS stets SCheck. Nur dann können Sie sicher sein, daß Ihre Eingaben korrekt sind.

(db)

Name	SCheck.bas
Sprache	GW-Basic und SBasic
Grafikkarte	jede
Besonderheiten	Checksummer für Listings in DOS International und DOS Extra

```
[SS5] 1ØØØØ '****************************************
      *******
[F29] 1ØØ1Ø '* SCheck - Prüfsummengenerator für DOS-Lis
      tings *
[ML4] 1ØØ2Ø '*                                       *
[8Ø7] 1ØØ3Ø '*            Konzeption: Volker Everts  *
[AC7] 1ØØ4Ø '*          Basic-Version: Markus Zietlow *
[XS5] 1ØØ5Ø '****************************************
      *******
[4UØ] 1ØØ6Ø '
[FL7] 1ØØ7Ø KEY OFF: DEFINT A-O,Q-Z: DEFSNG P: COLOR 7,
      Ø,1
[5G8] 1ØØ8Ø T = 1: TEMP = 2: F = 3'Ein-/Ausgabekanäle f
      estlegen
[332] 1ØØ9Ø FALSE = Ø
[9Ø2] 1Ø1ØØ TRUE = -1
[EY7] 1Ø11Ø ZEICHEN$ = "Ø123456789ABCDEFGHKLMNPRSTUVWXY
      Z"
[9H2] 1Ø12Ø LAUF = FALSE
[6B2] 1Ø13Ø BIN = FALSE
[851] 1Ø14Ø CLS
[9S1] 1Ø15Ø PRINT
[KØ7] 1Ø16Ø PRINT CHR$(2Ø1); STRING$(78, 2Ø5); CHR$(187
      );
[Z5C] 1Ø17Ø PRINT CHR$(186); STRING$(28, 32); "***  SCh
      eck 2.Ø2  ***"; STRING$(29, 32); CHR$(186);
[ØKE] 1Ø18Ø PRINT CHR$(186); " Basic-Version
                                (C) 1989 DOS Internatio
      nal "; CHR$(186);
[C5C] 1Ø19Ø PRINT CHR$(2Ø4); "═══════════════════════
      ═══════════════════════
      ═══"; CHR$(185);
[DR3] 1Ø2ØØ FOR ZAEHLER = 1 TO 5
[ZY6] 1Ø21Ø   PRINT CHR$(186); STRING$(78, 32); CHR$(18
      6);
[CR2] 1Ø22Ø NEXT ZAEHLER
[84C] 1Ø23Ø PRINT CHR$(2Ø4); "═══════════════════════
      ═══════════════════════
      ═══"; CHR$(185);
[ØØD] 1Ø24Ø PRINT CHR$(186); " CHECK:          ZEILE:
                          DATEI:
          "; CHR$(186);
[PØ7] 1Ø25Ø PRINT CHR$(2ØØ); STRING$(78, 2Ø5); CHR$(188
      );
[2D2] 1Ø26Ø LOCATE 2Ø, 1
[B17] 1Ø27Ø PRINT CHR$(218); STRING$(78, 196); CHR$(191
      );
[ØØ7] 1Ø28Ø PRINT CHR$(179); STRING$(78, 32); CHR$(179)
      ;
[C27] 1Ø29Ø PRINT CHR$(192); STRING$(78, 196); CHR$(217
      );
[4V1] 1Ø3ØØ COLOR 7
[9F1] 1Ø31Ø Y = 7
[9V6] 1Ø32Ø LOCATE 7, 31: PRINT "SCheck  Version 1.x"
[PR6] 1Ø33Ø LOCATE 8, 31: PRINT "SCheck  Version 2:Ø"
[M7H] 1Ø34Ø LOCATE 21, 4: PRINT "Bitte wählen Sie mit d
      en Pfeiltasten aus und bestätigen Sie durch [Ente
      r] ";
[2PD] 1Ø35Ø COLOR 14, 1: FOR Q = 3Ø TO 5Ø: LOCATE Y, Q:
       PRINT CHR$(SCREEN(Y, Q)): NEXT Q: COLOR 7, Ø
[UP6] 1Ø36Ø TASTE$ = INKEY$: IF TASTE$ = "" THEN 1Ø36Ø
[P96] 1Ø37Ø IF TASTE$ <> CHR$(Ø) + CHR$(72) THEN 1Ø4ØØ
[4ND] 1Ø38Ø IF Y <> 7 THEN FOR Q = 3Ø TO 5Ø: LOCATE Y,
      Q: PRINT CHR$(SCREEN(Y, Q)): NEXT Q: Y = Y - 1
[532] 1Ø39Ø GOTO 1Ø35Ø
[1A6] 1Ø4ØØ IF TASTE$ <> CHR$(Ø) + CHR$(8Ø) THEN 1Ø43Ø
[1ND] 1Ø41Ø IF Y <> 8 THEN FOR Q = 3Ø TO 5Ø: LOCATE Y,
      Q: PRINT CHR$(SCREEN(Y, Q)): NEXT Q: Y = Y + 1
[232] 1Ø42Ø GOTO 1Ø35Ø
[F35] 1Ø43Ø IF TASTE$ <> CHR$(13) THEN 1Ø36Ø
[8N2] 1Ø44Ø FOR Q = 7 TO 8
[U16] 1Ø45Ø   LOCATE Q, 3Ø: PRINT STRING$(21, 32)
[KN1] 1Ø46Ø NEXT Q
[8Y2] 1Ø47Ø VERSION = Y - 6
[GV1] 1Ø48Ø COLOR 7
[NF1] 1Ø49Ø Y = 7
[MA5] 1Ø5ØØ LOCATE 7, 36: PRINT "Standard"
[4A6] 1Ø51Ø LOCATE 8, 34: PRINT "Mit Prüfsumme"
[C65] 1Ø52Ø LOCATE 9, 37: PRINT "Hexdump"
[V7H] 1Ø53Ø LOCATE 21, 4: PRINT "Bitte wählen Sie mit d
      en Pfeiltasten aus und bestätigen Sie durch [Ente
      r] ";
[9PD] 1Ø54Ø COLOR 14, 1: FOR Q = 3Ø TO 5Ø: LOCATE Y, Q:
       PRINT CHR$(SCREEN(Y, Q)): NEXT Q: COLOR 7, Ø
[3R6] 1Ø55Ø TASTE$ = INKEY$: IF TASTE$ = "" THEN 1Ø55Ø
[4B6] 1Ø56Ø IF TASTE$ <> CHR$(Ø) + CHR$(72) THEN 1Ø59Ø
[BND] 1Ø57Ø IF Y <> 7 THEN FOR Q = 3Ø TO 5Ø: LOCATE Y,
      Q: PRINT CHR$(SCREEN(Y, Q)): NEXT Q: Y = Y - 1
[E32] 1Ø58Ø GOTO 1Ø54Ø
[GA6] 1Ø59Ø IF TASTE$ <> CHR$(Ø) + CHR$(8Ø) THEN 1Ø62Ø
[CND] 1Ø6ØØ IF Y <> 9 THEN FOR Q = 3Ø TO 5Ø: LOCATE Y,
      Q: PRINT CHR$(SCREEN(Y, Q)): NEXT Q: Y = Y + 1
[B32] 1Ø61Ø GOTO 1Ø54Ø
[P35] 1Ø62Ø IF TASTE$ <> CHR$(13) THEN 1Ø55Ø
[GN2] 1Ø63Ø FOR Q = 7 TO 9
[126] 1Ø64Ø   LOCATE Q, 3Ø: PRINT STRING$(21, 32)
[TN1] 1Ø65Ø NEXT Q
[WG4] 1Ø66Ø IF Y = 8 THEN LAUF = TRUE
[NA4] 1Ø67Ø IF Y = 9 THEN BIN = TRUE
[RU5] 1Ø68Ø LOCATE 12, 54: PRINT STRING$(25, 32)
[8DF] 1Ø69Ø GOSUB 1315Ø: LOCATE 21, 12: PRINT "Bitte ge
      ben Sie den Dateinamen der gewünschten Datei ein
      ";
[9P8] 1Ø7ØØ Y = 12: X = 54: LAENGE = 24: MENGE$ = "alle
      ": GOSUB 1Ø89Ø
[N6D] 1Ø71Ø IF (TASTE$ = CHR$(27)) OR (Q$ = "") THEN LO
      CATE 12, 54: PRINT STRING$(25, 32): GOTO 1Ø68Ø
[LM3] 1Ø72Ø ON ERROR GOTO 1116Ø
[C74] 1Ø73Ø OPEN Q$ FOR INPUT AS #T
[732] 1Ø74Ø GOTO 1123Ø
[3Ø2] 1Ø75Ø COLOR 12
[2A6] 1Ø76Ø LOCATE 8, 3Ø: PRINT "D A T E I F E H L E R"
[VV1] 1Ø77Ø COLOR 7
[5GC] 1Ø78Ø GOSUB 1315Ø: LOCATE 21, 21: PRINT "Drücken
      Sie bitte eine beliebige Taste ";
[B83] 1Ø79Ø WHILE INKEY$ = ""
[GX2] 1Ø8ØØ   PRINT CHR$(7);
[CE1] 1Ø81Ø WEND
[6U5] 1Ø82Ø LOCATE 8, 3Ø: PRINT STRING$(21, 32)
[842] 1Ø83Ø GOTO 1Ø68Ø
[2VØ] 1Ø84Ø '
[184] 1Ø85Ø '*********************************
[UD6] 1Ø86Ø '* Unterprogramm zur Dateneingabe *
[384] 1Ø87Ø '*********************************
[6VØ] 1Ø88Ø '
[GG1] 1Ø89Ø Q$ = ""
[EF2] 1Ø9ØØ LOCATE Y, X
[FS3] 1Ø91Ø PRINT Q$; CHR$(219);
[GR6] 1Ø92Ø TASTE$ = INKEY$: IF TASTE$ = "" THEN 1Ø92Ø
[ZK5] 1Ø93Ø IF (TASTE$ <> CHR$(8)) THEN 1Ø99Ø
[224] 1Ø94Ø   IF LEN(Q$) = Ø THEN 1Ø92Ø
[X44] 1Ø95Ø   Q$ = LEFT$(Q$, LEN(Q$) - 1)
[6N2] 1Ø96Ø   LOCATE Y, X
[V44] 1Ø97Ø   PRINT Q$; CHR$(219); " ";
[8A2] 1Ø98Ø   GOTO 1Ø92Ø
[T98] 1Ø99Ø IF TASTE$ = CHR$(13) THEN LOCATE Y, X: PRIN
      T Q$; " ";
[639] 11ØØØ IF (TASTE$ = CHR$(13)) OR (TASTE$ = CHR$(27
      )) THEN RETURN
[C55] 11Ø1Ø IF (ASC(TASTE$) < 32) THEN 1Ø92Ø
[NX4] 11Ø2Ø IF MENGE$ <> "zahl" THEN 11Ø5Ø
[ZC8] 11Ø3Ø   IF (ASC(TASTE$) < 48) OR (ASC(TASTE$) > 5
      7) THEN 1Ø92Ø
```

Listing. »SCheck.bas« für GW-Basic

```
[892] 11Ø4Ø   GOTO 11Ø7Ø
[PU4] 11Ø5Ø IF MENGE$ = "alle" THEN 11Ø7Ø
[E86] 11Ø6Ø IF INSTR(MENGE$, TASTE$) = Ø THEN 1Ø92Ø
[YF6] 11Ø7Ø IF LEN(Q$) < LAENGE THEN Q$ = Q$ + TASTE$
[T22] 11Ø8Ø GOTO 1Ø9ØØ
[6H8] 11Ø9Ø Y = 1Ø: X = 65: LAENGE = 5: MENGE$ = "zahl"
      : GOSUB 1Ø89Ø
[BV5] 111ØØ IF (TASTE$ = CHR$(27)) THEN RETURN
[6UØ] 11111Ø '
[NK2] 11112Ø '******************
[TR3] 11113Ø '* Fehler-Routine *
[RK2] 11114Ø '******************
[AUØ] 11115Ø '
[KN1] 11116Ø CLOSE
[ZG2] 11117Ø RESUME 1Ø75Ø
[DUØ] 11118Ø '
[8G2] 11119Ø '******************
[TH3] 112ØØ '* Hauptprogramm *
[4G2] 11211Ø '******************
[BUØ] 11122Ø '
[KP3] 11123Ø COLOR 7: GOSUB 1315Ø
[N43] 11124Ø ON ERROR GOTO Ø
[K62] 11125Ø DATEI$ = Q$
[DF7] 11126Ø GROSS$ = DATEI$: GOSUB 13Ø4Ø: DATEI$ = GROS
      S$
[3W3] 11127Ø I = INSTR(DATEI$, ".")
[6K7] 11128Ø IF I = Ø THEN DATEI$ = DATEI$ + ".": I = LE
      N(DATEI$)
[U64] 11129Ø EXT$ = MID$(DATEI$, I, 4)
[8T4] 113ØØ DATEI$ = LEFT$(DATEI$, I - 1)
[HN5] 11131Ø LOCATE 12, 54: PRINT DATEI$; EXT$
[HB4] 11132Ø TEMPNAME$ = DATEI$ + ".$$$"
[5E6] 11133Ø OPEN TEMPNAME$ FOR OUTPUT AS #TEMP
[CZ7] 11134Ø IF BIN THEN OPEN DATEI$ + ".BIN" FOR OUTPUT
       AS #F
[CP7] 11135Ø IF VERSION = 2 THEN NUMMER = 1 ELSE NUMMER
      = Ø
[KYC] 11136Ø IF BIN THEN GOSUB 1315Ø: LOCATE 21, 25: PRI
      NT ">>> Binärdatei wird erzeugt <<<"
[TE3] 11137Ø WHILE NOT EOF(T)
[X24] 11138Ø    LINE INPUT #T, ZEILE$
[LDB] 11385     WHILE RIGHT$(ZEILE$,1)=" " : ZEILE$=LEFT$
      (ZEILE$,LEN(ZEILE$)-1) : WEND
[333] 11139Ø    WEITER = FALSE
[WT3] 114ØØ     IF WEITER THEN 115ØØ
[WN6] 1141Ø     GOSUB 1182Ø 'Gelesene Zeile anzeigen
[TS4] 1142Ø     IF BIN THEN WEITER = TRUE
[ZT3] 1143Ø     IF WEITER THEN 115ØØ
[AM6] 1144Ø     CH$ = INKEY$: IF CH$ = "" THEN GOTO 1144Ø
[3G3] 1145Ø     ERLAUBT = FALSE
[ZH8] 1146Ø     IF CH$ = CHR$(13) THEN ERLAUBT = TRUE: WE
      ITER = TRUE
[N1C] 1147Ø     IF (CH$ = "e") OR (CH$ = "E") THEN ERLAUB
      T = TRUE: GOSUB 12Ø6Ø'Zeile bearbeiten
[8ME] 1148Ø     IF (CH$ = "q") OR (CH$ = "Q") THEN IF NOT
       LAUF THEN GOSUB 1242Ø: IF SCHLUSS THEN GOTO 1163
      Ø'Quit
[F57] 1149Ø     IF ERLAUBT THEN GOTO 114ØØ ELSE GOTO 1144
      Ø
[8Y3] 115ØØ     PRINT #TEMP, ZEILE$
[DU3] 1151Ø     IF NOT BIN THEN 1161Ø
[D99] 1152Ø     WHILE LEFT$(ZEILE$, 1) = " ": ZEILE$ = MI
      D$(ZEILE$, 2): WEND
[364] 1153Ø     IF ZEILE$ = "" THEN 1161Ø
[K75] 1154Ø     B = VAL("&H" + LEFT$(ZEILE$, 2))
[TSH] 1155Ø     IF (B < Ø) OR (B > 255) THEN GOSUB 1315Ø:
       LOCATE 21, 22: PRINT "+++++ ABBRUCH WEGEN DATENF
      EHLER +++++": LOCATE 25, 1: END
[HN1] 1156Ø     BY = B
[TK3] 1157Ø     PRINT #F, CHR$(BY);
[YA4] 1158Ø     ZEILE$ = MID$(ZEILE$, 3)
[M99] 1159Ø     WHILE LEFT$(ZEILE$, 1) = " ": ZEILE$ = MI
      D$(ZEILE$, 2): WEND
[T92] 116ØØ     GOTO 1153Ø
[DY3] 1161Ø     NUMMER = NUMMER + 1
[8E1] 1162Ø WEND
[S7B] 1163Ø WHILE NOT EOF(T): LINE INPUT #T, ZEILE$: PR
      INT #TEMP, ZEILE$: WEND
[4P1] 1164Ø CLOSE
[8U5] 1165Ø OPEN DATEI$ + EXT$ FOR INPUT AS #1
[XØ6] 1166Ø OPEN DATEI$ + ".BAK" FOR OUTPUT AS #2
[XGB] 1167Ø WHILE NOT EOF(1): LINE INPUT #1, ZEILE$: PR
      INT #2, ZEILE$: WEND: CLOSE
[6D5] 1168Ø OPEN TEMPNAME$ FOR INPUT AS #1
[576] 1169Ø OPEN DATEI$ + EXT$ FOR OUTPUT AS #2
[UGB] 117ØØ WHILE NOT EOF(1): LINE INPUT #1, ZEILE$: PR
      INT #2, ZEILE$: WEND: CLOSE
[713] 1171Ø KILL TEMPNAME$
[GTN] 1172Ø GOSUB 1256Ø: LOCATE 12, 9: PRINT STRING$(11
      , 32): LOCATE 12, 25: PRINT STRING$(21, 32): LOCA
      TE 12, 53: PRINT STRING$(27, 32)'Statuszeile lösc
      hen
[494] 1173Ø IF BIN THEN EXT$ = ".BIN"
[N4H] 1174Ø GOSUB 1315Ø: LOCATE 21, (8Ø - (31 + LEN(DAT
      EI$ + EXT$))) / 2: PRINT "Die Datei '"; DATEI$ +
      EXT$; "' wurde gespeichert."
[CE2] 1175Ø LOCATE 25, 1
[H51] 1176Ø END
[4VØ] 1177Ø '
[TH3] 1178Ø '****************************
[B65] 1179Ø '* Gelesene Zeile anzeigen *
[NH3] 1180Ø '****************************
[2VØ] 1181Ø '
[BBB] 1182Ø IF LAUF THEN CHECK$ = "[" + LEFT$(ZEILE$, 3
      ) + "]": ZEILE$ = MID$(ZEILE$, 5)
[MDN] 1183Ø FOR Q = 1 TO LEN(CHECK$): IF ASC(MID$(CHECK
      $, Q, 1)) > 96 AND ASC(MID$(CHECK$, Q, 1)) < 123
      THEN MID$(CHECK$, Q, 1) = CHR$(ASC(MID$(CHECK$, Q
      , 1)) - 32)
[3P1] 1184Ø NEXT Q
[B57] 1185Ø LINKS = 2: OBEN = 6: RECHTS = 79: UNTEN = 1
      Ø
[CX4] 1186Ø LOCATE 12, 26: PRINT NUMMER
[Z45] 1187Ø GOSUB 1256Ø'Fenster löschen
[WX8] 1188Ø GOSUB 1266Ø'Zeile innerhalb des Fensters an
      zeigen
[366] 1189Ø GOSUB 128ØØ'Check-Summe errechnen
[ØM2] 119ØØ LOCATE 12, 1Ø
[HB2] 1191Ø PRINT PS$
[HM7] 1192Ø IF BIN THEN GOTO 12ØØØ'Binärdatei erzeugen
[N8C] 1193Ø IF LAUF THEN GOTO 1197Ø'Übereinstimmung der
       Prüfsummen feststellen
[X74] 1194Ø GOSUB 1315Ø: LOCATE 21, 4
[LKD] 1195Ø PRINT "[E] Editieren    [RETURN] Weiter
      [Q] Quit       [Ctrl-Break] Abbrechen"
[152] 1196Ø GOTO 1199Ø
[ATP] 1197Ø GOSUB 1315Ø: IF PS$ = CHECK$ THEN LOCATE 21
      , (8Ø - (22 + LEN(STR$(NUMMER)))) / 2: PRINT "Prü
      fsumme OK für Zeile"; NUMMER: WEITER = TRUE: GOTO
       12ØØØ
[1NH] 1198Ø LOCATE 21, 4: PRINT CHR$(7); "Prüfsummenfeh
      ler!  [E] Editieren  [RETURN] Weiter  [Ctrl-Break
      ] Abbrechen"
[XM2] 1199Ø LOCATE 12, 15
[KV1] 12ØØØ RETURN
[5UØ] 12Ø1Ø '
[MK2] 12Ø2Ø '******************
[4H3] 12Ø3Ø '* Zeile eingeben *
[PK2] 12Ø4Ø '******************
[9UØ] 12Ø5Ø '
[HRD] 12Ø6Ø GOSUB 1315Ø: LOCATE 21, 17: PRINT "Ändern S
      ie die Programmzeile        [Enter] Ende"
[245] 12Ø7Ø GOSUB 1256Ø'Fenster löschen
[7B5] 12Ø8Ø GOSUB 1266Ø'Zeile darstellen
[FL3] 12Ø9Ø LOCATE OBEN, LINKS
[7G3] 121ØØ X = 1: PRINT "■";
[8E2] 1211Ø CH$ = INKEY$
[FE6] 1212Ø IF LEFT$(CH$, 1) <> CHR$(Ø) THEN GOTO 1222Ø
[4N3] 1213Ø   CH$ = RIGHT$(CH$, 1)
[4NE] 1214Ø   IF CH$ = "M" THEN IF X > LEN(ZEILE$) THEN
       PRINT CHR$(7); : GOTO 1211Ø ELSE GOSUB 1234Ø: X
      = X + 1
[KZC] 1215Ø   IF CH$ = "K" THEN IF X = 1 THEN PRINT CHR
      $(7); : GOTO 1211Ø ELSE GOSUB 1234Ø: X = X - 1
[YZC] 1216Ø   IF CH$ = "H" THEN IF X < 79 THEN PRINT CH
      R$(7); : GOTO 1211Ø ELSE GOSUB 1234Ø: X = X - 78
[YCF] 1217Ø   IF CH$ = "P" THEN IF X + 77 > LEN(ZEILE$)
       THEN PRINT CHR$(7); : GOTO 1211Ø ELSE GOSUB 1234
      Ø: X = X + 78
[TV7] 1218Ø   IF CH$ = "G" THEN IF X <> 1 THEN GOSUB 12
      34Ø: X = 1
[41C] 1219Ø   IF CH$ = "O" THEN IF X <> LEN(ZEILE$) + 1
       THEN GOSUB 1234Ø: X = LEN(ZEILE$) + 1
[8PM] 122ØØ   IF CH$ = "S" THEN IF X <= LEN(ZEILE$) THE
      N ZEILE$ = LEFT$(ZEILE$, X - 1) + MID$(ZEILE$, X
      + 1): GOSUB 1266Ø ELSE PRINT CHR$(7); : GOTO 121
      Ø
[G92] 1221Ø   GOTO 1224Ø
[XXN] 1222Ø IF CH$ = CHR$(8) THEN IF X > 1 THEN ZEILE$
      = LEFT$(ZEILE$, X - 2) + MID$(ZEILE$, X): GOSUB 1
      266Ø: GOSUB 1234Ø: X = X - 1 ELSE PRINT CHR$(7);
      : GOTO 1211Ø
[LZN] 1223Ø IF CH$ >= " " THEN IF LEN(ZEILE$) = 255 THE
      N PRINT CHR$(7); : GOTO 1211Ø ELSE ZEILE$ = LEFT$
      (ZEILE$, X - 1) + CH$ + MID$(ZEILE$, X): GOSUB 12
      66Ø: X = X + 1
[7YB] 1224Ø LOCATE OBEN + INT((X - 1) / 78), LINKS - 1
      + X - INT((X - 1) / 78) * 78: PRINT "■";
[LK5] 1225Ø IF CH$ <> CHR$(13) THEN GOTO 1211Ø
[648] 1226Ø IF LAUF THEN ZEILE$ = MID$(CHECK$, 2, 3) +
      " " + ZEILE$
[XL6] 1227Ø GOSUB 1234Ø: GOSUB 1266Ø'Zeile anzeigen
[2W1] 1228Ø RETURN
[NUØ] 1229Ø '
[CP3] 123ØØ '****************************
[2H5] 1231Ø '* Zeichen wiederherstellen *
[EP3] 1232Ø '****************************
```

Listing. »SCheck.bas« für GW-Basic

```
[LUØ] 1233Ø '
[UM9] 1234Ø LOCATE OBEN + INT((X - 1) / 78), LINKS - 1
      + X - INT((X - 1) / 78) * 78
[YXA] 1235Ø IF MID$(ZEILE$, X, 1) = "" THEN PRINT " " E
      LSE PRINT MID$(ZEILE$, X, 1)
[4W1] 1236Ø RETURN
[RUØ] 1237Ø '
[ØA2] 1238Ø '***************
[3X2] 1239Ø '* Quit (Ende) *
[W92] 1240Ø '***************
[NUØ] 1241Ø '
[U63] 1242Ø SCHLUSS = FALSE
[KYB] 1243Ø SICHER$ = "": FOR Q = 2 TO 79: SICHER$ = SI
      CHER$ + CHR$(SCREEN(21, Q)): NEXT Q
[YDD] 1244Ø GOSUB 1315Ø: LOCATE 21, 16: PRINT "Bitte dr
      ücken Sie [J], um die Arbeit zu beenden "
[GA6] 1245Ø CH$ = INKEY$: IF CH$ = "" THEN GOTO 1245Ø
[R3B] 1246Ø IF (CH$ = "j") OR (CH$ = "J") THEN SCHLUSS
      = TRUE: PRINT #TEMP, ZEILE$
[BA2] 1247Ø GOSUB 1315Ø
[P78] 1248Ø IF NOT SCHLUSS THEN LOCATE 21, 2: PRINT SIC
      HER$
[BW1] 1249Ø RETURN
[651] 1250Ø END
[TUØ] 1251Ø '
[6R2] 1252Ø '*******************
[U24] 1253Ø '* Fenster löschen *
[8R2] 1254Ø '*******************
[XUØ] 1255Ø '
[Ø44] 1256Ø FOR Q = OBEN TO UNTEN
[Z53] 1257Ø LOCATE Q, LINKS
[8B6] 1258Ø   PRINT STRING$(RECHTS - LINKS + 1, 32)
[ZN1] 1259Ø NEXT Q
[AW1] 1260Ø RETURN
[XUØ] 1261Ø '
[265] 1262Ø '*****************************************
      *
[D98] 1263Ø '* Inhalt von »zeile$« im Fenster anzeigen
      *
[465] 1264Ø '*****************************************
      *
[1VØ] 1265Ø '
[SR2] 1266Ø F.X = Ø: F.Y = Ø
[CC5] 1267Ø FOR ZAEHLER = 1 TO LEN(ZEILE$)
[F85] 1268Ø   LOCATE OBEN + F.Y, LINKS + F.X
[UH5] 1269Ø   PRINT MID$(ZEILE$, ZAEHLER, 1);
[SK2] 1270Ø   F.X = F.X + 1
[597] 1271Ø   IF F.X >= RECHTS - 1 THEN F.X = Ø: F.Y =
      F.Y + 1
[6S2] 1272Ø NEXT ZAEHLER
[RX6] 1273Ø LOCATE OBEN + F.Y, LINKS + F.X: PRINT " ";
[KW1] 1274Ø RETURN
[5VØ] 1275Ø '
[AH4] 1276Ø '************************************
[ET6] 1277Ø '* Zeile auf Check-Summe überprüfen *
[CH4] 1278Ø '************************************
```

```
[9VØ] 1279Ø '
[F42] 128ØØ P = Ø: A = 1
[8SB] 1281Ø WHILE (MID$(ZEILE$, A, 1) = " ") OR (MID$(Z
      EILE$, A, 1) = CHR$(9)): A = A + 1: WEND
[374] 1282Ø FOR I = A TO LEN(ZEILE$)
[4CX] 1283Ø   IF MID$(ZEILE$, I, 1) >= " " THEN GROSS$
      = MID$(ZEILE$, I, 1): IF VERSION = 1 THEN P = ((I
       - A + 1) AND 7) * ASC(GROSS$) + P: GOTO 1292Ø EL
      SE GOSUB 13Ø4Ø: P = (((I - A + 1) AND 3) + 1) * A
      SC(GROSS$) + P: GOTO 1292Ø
[S22] 1284Ø   COLOR 12
[5N6] 1285Ø   LOCATE 8, 3Ø: PRINT "D A T E I F E H L E
      R"
[Ø22] 1286Ø   COLOR 7
[X7C] 1287Ø   LOCATE 21, 11: PRINT "          Drücken S
      ie bitte eine beliebige Taste         ";
[CC3] 1288Ø   WHILE INKEY$ = ""
[D93] 1289Ø     PRINT CHR$(7);
[5L1] 129ØØ   WEND
[R16] 1291Ø   LOCATE 8, 3Ø: PRINT STRING$(21, 32)
[ØP1] 1292Ø NEXT I
[KF2] 1293Ø PS$ = "[    ]"
[AM2] 1294Ø FOR I = 2 TO 4
[K25] 12945   IF P > 32767 THEN P = P - 32768
[5T7] 1295Ø   MID$(PS$, I, 1) = MID$(ZEICHEN$, (P AND 3
      1) + 1, 1)
[HZ2] 1296Ø   P = FIX(P / 32)
[5P1] 1297Ø NEXT I
[YW1] 1298Ø RETURN
[HVØ] 1299Ø '
[ZX3] 13ØØØ '*********************************
[KØ6] 13Ø1Ø '* In Großbuchstaben umwandeln *
[934] 13Ø2Ø '*********************************
[AUØ] 13Ø3Ø '
[9ZN] 13Ø4Ø FOR Q = 1 TO LEN(GROSS$): IF ASC(MID$(GROSS
      $, Q, 1)) > 96 AND ASC(MID$(GROSS$, Q, 1)) < 123
      THEN MID$(GROSS$, Q, 1) = CHR$(ASC(MID$(GROSS$, Q
      , 1)) - 32)
[8T8] 13Ø5Ø IF MID$(GROSS$, Q, 1) = "ä" THEN MID$(GROSS
      $, Q, 1) = "Ä"
[NU8] 13Ø6Ø IF MID$(GROSS$, Q, 1) = "ö" THEN MID$(GROSS
      $, Q, 1) = "Ö"
[UU8] 13Ø7Ø IF MID$(GROSS$, Q, 1) = "ü" THEN MID$(GROSS
      $, Q, 1) = "Ü"
[DN1] 13Ø8Ø NEXT Q
[YV1] 13Ø9Ø RETURN
[BUØ] 131ØØ '
[YZ2] 1311Ø '**********************
[9N4] 1312Ø '* Meldezeile löschen *
[ØØ3] 1313Ø '**********************
[FUØ] 1314Ø '
[ZT5] 1315Ø LOCATE 21, 2: PRINT STRING$(78, 32)
[ZV1] 1316Ø RETURN
```

Listing. »SCheck.bas« für GW-Basic

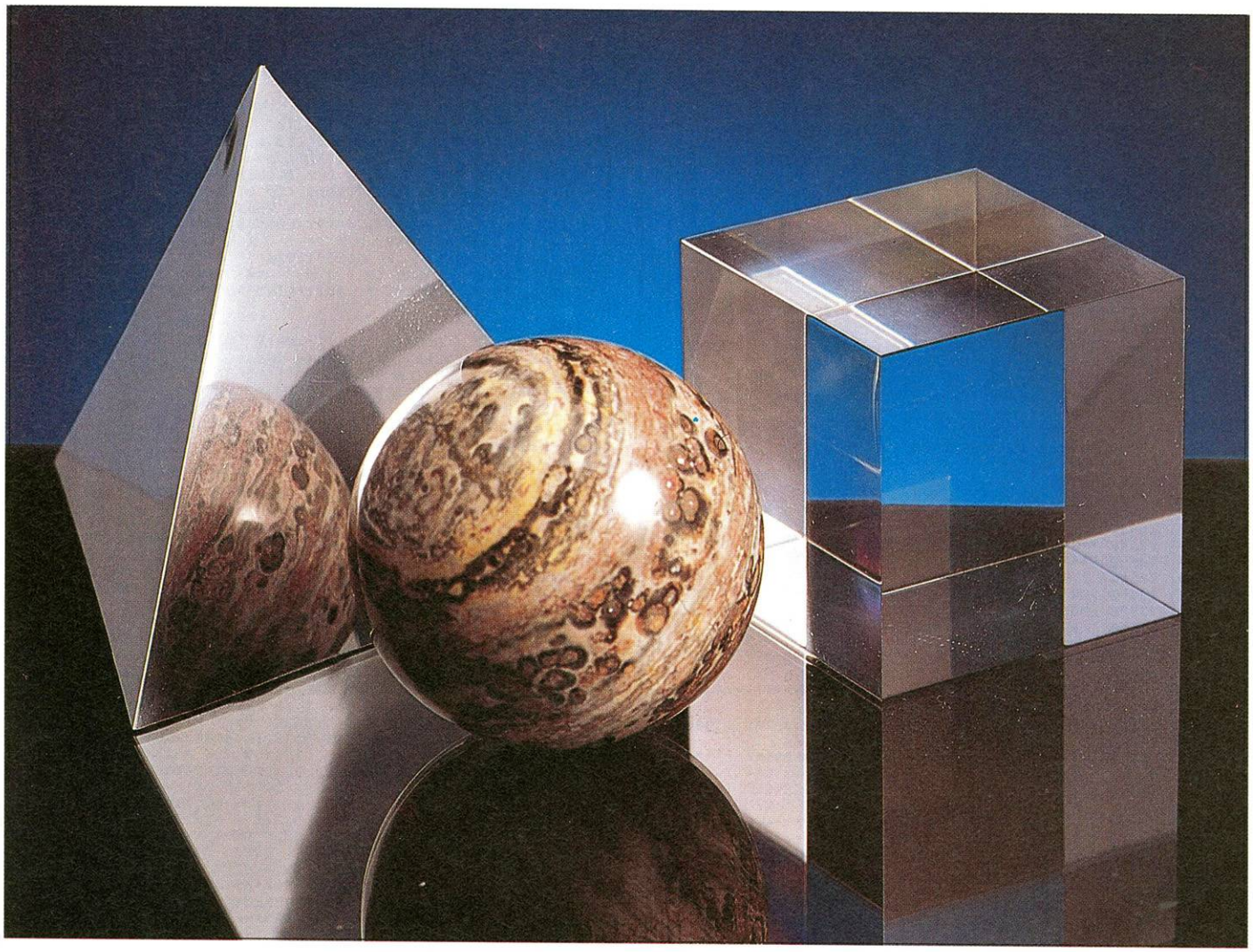
Toolboxen für Programmierer gesucht

Haben Sie eine Toolbox in C, Pascal, Basic, dBase, Clipper oder Assembler geschrieben? Beispielsweise schnelle, universelle Grafikroutinen, eine Textverarbeitungs- oder eine Datenbank-Toolbox? Dann sollten Sie sich schnellstmöglich mit uns in Verbindung setzen. Schicken Sie uns unverbindlich eine Diskette mit Ihrem Programm und einen Ausdruck der Dokumentation.

Falls Sie sich vorweg informieren möchten: Zwischen 15.00 und 17.00 Uhr stehen wir Ihnen an allen Werktagen unter der Rufnummer (0 81 21) 7 69-0 zur Verfügung.

Schicken Sie Ihre Unterlagen an folgende Adresse:

Redaktion DOS International, »DOS Software«,
Gruberstraße 46c, 8011 Poing



Turbo Pascal ab Version 5.0

Vom Raum auf den Bildschirm

Wen haben nicht schon die dreidimensionalen Rotationskörper in Demo-Programmen fasziniert. Dabei taucht natürlich immer die Frage auf: »Wie machen die das?« Wir stellen Ihnen eine Unit vor, mit der auch Sie in der Lage sind, sich bewegende räumliche Vektorgrafiken zu generieren.

Um perspektivische 3D-Objektdarstellungen mit weniger Aufwand zu programmieren, verwenden Sie die Unit aus »simple3d.pas« (Listing 1): Ein kurzes Turbo-Pascal-Programm, das mit Hilfe des BGI (Borland Graphic Interface) drei Buchstaben im Raum zum Rotieren bringt.

Den Programmkern stellen dabei die drei kurzen Prozeduren »Rotation3D«, »Calc3D« und »Window3D« dar. Diese sind in der Unit zusammengefaßt. Sie rechnen einen Raumpunkt zur Darstellung auf den Bildschirm um.

Die drei Prozeduren der Unit benötigen zusätzliche Informationen in den folgenden globalen Variablen: »mx«, »my« und »mz« geben die räumliche Position des Betrachters an. Die Variable »d« bestimmt den perspektivischen Faktor. Die Prozeduren rechnen stets nur einen Punkt um. Dazu legen Sie zunächst mit den drei Parametern »r1«, »r2«, »r3« der Prozedur »Rotation3D« die Lage des Koordinatensystems im Raum fest. Jetzt folgt die eigentliche Punktberechnung mit der Prozedur »Calc3D«. Als Parameter übergeben Sie ihr die x-, y- und z-Koordinate des Punktes. Anschließend rechnen Sie mit »Window3D« den Punkt auf die Bildschirmausmaße um. Die Bildschirmkoordinaten liegen daraufhin in den globalen Variablen »sx« und »sy«. Das Demo-Programm »demo3d.pas« (Listing 2) zeigt Ihnen die Verwendung der Unit.

(Martin Backschat/hs)

```
[WR3]    1: { Programm: demo3d.pas
[315]    2:   Funktion: Buchstabenrotation
[YY5]    3:   Sprache:  Turbo Pascal ab Version 5.Ø
[774]    4:   Autor:    Martin Backschat
[P35]    5:   (C)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    6:
[E44]    7: USES Graph,simple3d,Crt;
[ØØØ]    8:
[6XØ]    9: CONST
[ER1]   1Ø:   lines = 12;
[WX3]   11:   dx = Ø.Ø5; dy = Ø.Ø7; dz = -2.1;
[TP4]   12:   dr1 = Ø.Ø1; dr2 = Ø.Ø12; dr3 = Ø.Ø18;
[ØØØ]   13:
[STØ]   14: TYPE
[YY6]   15:   pointarray = ARRAY[1..lines] OF INTEGER;
[ØØØ]   16:
[6XØ]   17: CONST
[WH5]   18:   xs : pointarray = (-2ØØ,-2ØØ,-2ØØ,-5Ø,5Ø,
[NC3]   19:                      -5Ø,-5Ø,1ØØ,1ØØ,1ØØ,2ØØ,1ØØ);
[9E5]   2Ø:   xe : pointarray = (-2ØØ,-1ØØ,-1ØØ,5Ø,5Ø,
[963]   21:                      5Ø,-5Ø,2ØØ,1ØØ,2ØØ,2ØØ,2ØØ);
[PE5]   22:   ys : pointarray = (1ØØ,1ØØ,-1ØØ,1ØØ,1ØØ,
[863]   23:                      -1ØØ,-1ØØ,1ØØ,1ØØ,Ø,Ø,-1ØØ);
[H15]   24:   ye : pointarray = (-1ØØ,Ø,Ø,1ØØ,-1ØØ,
[663]   25:                      -1ØØ,1ØØ,1ØØ,Ø,Ø,-1ØØ,-1ØØ);
[LH5]   26:   zk : pointarray = (Ø,Ø,Ø,-5Ø,-5Ø,-5Ø,-5Ø,
[4Ø3]   27:                       -1ØØ,-1ØØ,-1ØØ,-1ØØ,-1ØØ);
[RNØ]   28: VAR
[U45]   29:   maxx,minx,maxy,miny : INTEGER;
[RT4]   3Ø:   ox1,ox2,oy1,oy2 : pointarray;
[1V2]   31:   i,x1,y1 : INTEGER;
[MS6]   32:   r1,r2,r3 : real;  { Krd-System-Rotation }
[ØØØ]   33:
[ZH7]   34: {Initialisiert das BGI vorschriftsmaessig}
[D43]   35: PROCEDURE initbgi;
[176]   36: VAR error,grdriver,grmode : INTEGER;
[MUØ]   37: BEGIN
[W93]   38:   grdriver := detect;
[U25]   39:   InitGraph(grdriver,grmode,'?');
[467]   4Ø: { Bitte anstelle '?' Pfad der BGI-Treiber }
[2M3]   41:   error := GraphResult;
[LF4]   42:   IF error <> grok THEN BEGIN
[6E4]   43:     WriteLn('Graphics error:',
[XU3]   44:         GraphErrorMsg(error));
[K96]   45:     ReadLn; Halt(1); {Ende mit Fehlercode}
[ZNØ]   46:   END;
[AZ5]   47:   screen_x := GetMaxX-1;  { Wichtig! }
[FZ5]   48:   screen_y := GetMaxY-1;  { Wichtig! }
[L24]   49:   maxx := Round(screen_x);
[UA4]   5Ø:   maxy := Round(screen_y)+1;
[ZNØ]   51: END;
[ØØØ]   52:
[997]   53: { Demonstration der drei 3-D Funktionen! }
[MUØ]   54: BEGIN
[WK5]   55:   initbgi;  { BGI-Treiber init. }
[SX3]   56:   SetColor(GetMaxColor);
[ØW3]   57:   SetWriteMode(xorput);
[NV3]   58:   r1 := Ø.Ø; r2 := Ø.Ø; r3 := Ø.Ø;
[N74]   59:   FOR i := 1 TO lines DO BEGIN
[5U5]   6Ø:     ox1[i] := Ø; oy1[i] := Ø; { Init. }
[UT3]   61:     ox2[i] := Ø; oy2[i] := Ø;
[ZNØ]   62:   END;
[2Y4]   63:   WHILE NOT KeyPressed DO BEGIN
[N74]   64:     FOR i := 1 TO lines DO BEGIN
[CB3]   65:       rotation3d(r1,r2,r3);
[6G4]   66:       calc3d(xs[i],ys[i],zk[i]);
[EH1]   67:       window3d;
[5W2]   68:       x1 := sx; y1 := sy;
[KD4]   69:       calc3d(xe[i],ye[i],zk[i]);
[EH1]   7Ø:       window3d;
[616]   71:       Line(ox1[i],oy1[i],ox2[i],oy2[i]);
[594]   72:       ox1[i] := x1; oy1[i] := y1;
[AD4]   73:       ox2[i] := sx; oy2[i] := sy;
[WU2]   74:       Line(x1,y1,sx,sy);
[ZNØ]   75:     END;
[1T5]   76:     mx := mx+dx; my := my+dy; mz := mz+dz;
[TK5]   77:     r1 := r1+dr1; r2 := r2+dr2; r3:=r3+dr3;
[ZNØ]   78:   END;
[522]   79:   CloseGraph;
[KNØ]   8Ø: END.
```

Listing 1. Die Unit »simple3d.pas« enthält alle wichtigen Prozeduren zum Darstellen von 3D-Objekten auf dem Bildschirm

```
[C44]    1: { Programm: simple3d.pas
[L34]    2:   Funktion: 3D-Prozeduren
[YY5]    3:   Sprache:  Turbo Pascal ab Version 5.Ø
[774]    4:   Autor:    Martin Backschat
[P35]    5:   (C)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    6:
[UD2]    7: UNIT simple3d;
[ØØØ]    8:
[SK1]    9: INTERFACE
[ØØØ]   1Ø:
[RNØ]   11: VAR
[T77]   12:   sx,sy : INTEGER;  {Bildschirmkoordinaten}
[5V6]   13:   mx,my,mz : real;  {Betrachterposition   }
[MV6]   14:   d : real;         {Vergrößerungsfaktor  }
[AM7]   15:   screen_x, screen_y : real;  {Scr.ausmaße}
[ØØØ]   16:
[AE5]   17: PROCEDURE calc3d(xk,yk,zk : real);
[M16]   18: PROCEDURE rotation3d(r1,r2,r3 : real);
[TD3]   19: PROCEDURE window3d;
[ØØØ]   2Ø:
[4L2]   21: IMPLEMENTATION
[6XØ]   22: CONST
[8A6]   23:   world_width = 1ØØØ;   { Groesse des }
[WB6]   24:   world_height = 1ØØØ;  { Weltsystems }
[RNØ]   25: VAR
[NH3]   26:   x,y,z,xa,ya,za : real;
[1S5]   27:   sr1,sr2,sr3,cr1,cr2,cr3,rx,ry : real;
[ØØØ]   28:
[EW6]   29: { Berechnet Rotationswerte fuer Calc3D }
[M16]   3Ø: PROCEDURE rotation3d(r1,r2,r3 : real);
[MUØ]   31: BEGIN
[G86]   32:   sr1:=Sin(r1); sr2:=Sin(r2); sr3:=Sin(r3);
[N36]   33:   cr1:=Cos(r1); cr2:=Cos(r2); cr3:=Cos(r3);
[ZNØ]   34: END;
[ØØØ]   35:
[WT6]   36: { Berechnet die 2-D-Koordinaten aus den
[767]   37: 3-D Weltkoordinaten (x,y,z), die zuvor noch
[UB7]   38: mit den Werten von Rotation3D rot. werden }
[ØØØ]   39:
[AE5]   4Ø: PROCEDURE calc3d(xk,yk,zk : real);
[MUØ]   41: BEGIN
[9X3]   42:   x := -xk; y := yk; z := zk;
[EA5]   43:   xa := cr1*x-sr1*z; za := sr1*x+cr1*z;
[AL5]   44:   x := cr2*xa+sr2*y; ya := cr2*y-sr2*xa;
[323]   45:   z := cr3*za-sr3*ya;
[MK4]   46:   y := sr3*za+cr3*ya; z := z+mz;
[PL4]   47:   x := d*(x+mx)/z; y := d*(y+my)/z;
[ZNØ]   48: END;
[ØØØ]   49:
[5M7]   5Ø: { Rechnet die Bildschirmkoordinaten aus den
[UG7]   51: 2-D-Koordinaten und legt sie in (sx,sy) ab}
[TD3]   52: PROCEDURE window3d;
[MUØ]   53: BEGIN
[BZ3]   54:   x := x+(world_width/2-1);
[624]   55:   y := y+(world_height/2-1);
[1B5]   56:   rx := screen_x/(world_width-1);
[M95]   57:   ry := screen_y/(world_height-1);
[PV2]   58:   sx := Round(x*rx);
[WV2]   59:   sy := Round(y*ry);
[ZNØ]   6Ø: END;
[ØØØ]   61:
[MUØ]   62: BEGIN
[EE1]   63:   d := 75Ø.Ø;
[464]   64:   mx := Ø.Ø; my := Ø.Ø; mz := -35Ø;
[4Ø4]   65:   xa := Ø.Ø; ya := Ø.Ø; za := Ø.Ø;
[5R4]   66:   sr1 := Ø.Ø; sr2 := Ø.Ø; sr3 := Ø.Ø;
[5M4]   67:   cr1 := Ø.Ø; cr2 := Ø.Ø; cr3 := Ø.Ø;
[PL2]   68:   rx := Ø.Ø; ry := Ø.Ø;
[KNØ]   69: END.
```

Listing 2. 3D-Grafik muß nicht kompliziert sein: »demo3d.pas« rotiert die Buchstaben »D O S« im Raum

Turbo Pascal ab Version 4.0

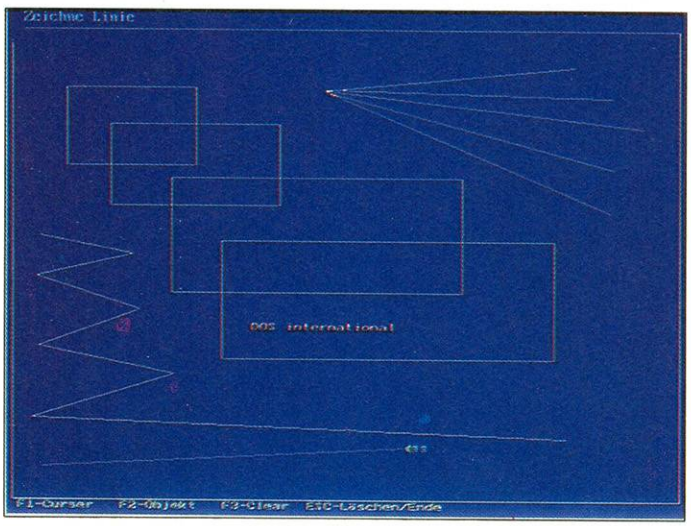
Schnelle Grafiken

Viele Anwender programmieren in Turbo Pascal, weil mit der Unit »Graph« leistungsfähige Grafikroutinen zur Verfügung stehen, womit sich auch anspruchsvolle Aufgaben oder größere Projekte realisieren lassen. Wenn Sie Besitzer einer EGA/VGA-Karte sind, geben wir Ihnen Tips, wie Sie Grafikprogramme erzeugen können, die auch höheren Ansprüchen genügen.

Mancher Grafikprogrammierer wundert sich über die Funktionsweise schneller Bildschirmroutinen bei professionellen Anwendungen. Diese benutzen zum Erzeugen, Identifizieren oder Editieren grafischer Elemente optische Hilfsmittel wie Fadenkreuz, Gummiband, dehnbare Rechtecke oder Hilfsrahmen, die sich mit der Maus über den Bildschirm ziehen oder verschieben lassen und anschließend wieder ausgeblendet werden. Solche Funktionen lassen sich auch mit Turbo Pascal leicht generieren (Bild).

Konstruktionshilfen dieser Art bestehen meist aus einer oder mehreren Linien, die mittels Xor-Modus mit dem Inhalt des Bildschirmspeichers verknüpft werden, das heißt, wenn für einen relevanten Bildpunkt ein Farbattribut, beispielsweise der Rot-Anteil, gesetzt ist, wird es ausgeblendet. Im umgekehrten Fall werden nicht gesetzte Komponenten eingeblendet. Die Farbgebung einer so erzeugten Linie ist somit nicht für alle Bildpunkte gleich, sondern jeweils komplementär zur ursprünglichen Farbgebung. Die Linie wirkt dadurch optisch transparent, vergleichbar einem Farbfilter. Der Hintergrund bleibt erkennbar und rekonstruierbar, denn durch die Umkehrung, die man mit der Wiederholung des Vorgangs erreicht, verschwindet die Linie, und der alte Zustand ist wiederhergestellt.

Läßt man nun solche Operationen in schneller Folge ablaufen, kann man bewegliche, transparente Elemente erzeugen. Dabei spielt die Zeit eine entscheidende Rolle, denn es müssen blitzschnell eine erhebliche Anzahl von Bildpunktpositionen berechnet, ihre Farbwerte gelesen und geändert werden – und alles für das Auge möglichst trägheitsfrei der Mausbewegung folgend. Diese Aufgabe überlassen Sie am einfachsten weiterhin Turbo Pascal oder der Hardware.



Ohne großen Aufwand sind professionelle Grafiktechniken in Turbo Pascal zu verwirklichen

Exklusive Grafiken mit der Exclusive-Oder-Verknüpfung

In der Unit Graph suchen Sie aber zunächst vergeblich nach einem Modus, eine »Xor-Linie« zu erzeugen. Der Xor-Modus läßt sich bei den Prozeduren »Line«, »LineTo« und »LineRel« nicht optional festlegen. Der Bildschirm wird immer mit der aktuellen Zeichenfarbe überschrieben, womit die Information über den ursprünglichen Zustand im Bildschirmspeicher verloren geht. Es liegt daher nahe, selbst den gewünschten Verknüpfungsmodus anzuwählen. Dazu wird unmittelbar vor Aufruf der Prozedur »Line« das Data-Rotate-Register der Graphikkarte angesprochen und die Xor-Verknüpfung eingestellt. Entscheidend ist nun, daß sich Turbo Pascal angesichts dieser kleinen Manipulation gutmütig verhält und nicht alles sofort wieder rückgängig macht. Allerdings ist eine einmalige Anweisung dieser Art zu wenig, da nach Erledigung der Aufgabe die Register ihre Standardwerte erhalten. Mit Einführung der kleinen Prozedur »LineXor«:

```
procedure
LineXor(x1,y1,x2,y2:integer);
begin
Port[$3CE]:=3; {Registerwahl}
Port[$3CF]:=$18; {Xor-Modus}
Line(x1,y1,x2,y2); {Linie}
end;
```

gelangt man problemlos und ohne Zeitverlust ans Ziel, das heißt, fast – denn bei durchgezogenen horizontalen Linien funktioniert der Xor-Put nicht. Diese werden offensichtlich gesondert behandelt. Am besten wählt man vorab eine benutzerdefinierte Linienart, zum Beispiel die mit dem schönen Bitmuster Dez 43690 (feinstens gepunktet), was den Hilfslinien-charakter noch betont. Nun läuft alles nach Wunsch, und selbst erzeugte optische Hilfen können im professionellen Gewand über den Bildschirm gleiten.

Wer bewegliche Rechtecke kreieren möchte, sollte erst gar nicht versuchen, mit der Prozedur »rectangle« entsprechend zu verfahren, denn bei Turbo Pascal besteht ein Rechteck auch nur aus vier Linien. Erzeugen Sie das Rechteck mittels Xor-Linien, oder noch besser, fassen Sie die vier Linexor-Aufrufe zu einer Prozedur »RectangleXor« zusammen. So sparen Sie Laufzeit ein und bekommen keine Probleme.

Für alle, die sich noch mehr optische Effekte wünschen, ein weiterer Tip. Das Beispiel mit dem Rechteck lehrt Sie, daß bei Turbo-Pascal-Prozeduren, die ihrerseits Unterroutinen aufrufen, der Trick mit der Xor-Anweisung nutzlos ist, da sie spätestens vor Aufruf der ersten internen Unterroutine überschrieben wird, wie bei Kreisen oder Ellipsen.

Sie können jedoch Schriften in die Grafik einbinden. Dazu dient die Prozedur »outtextxy«, die dazu den bitmusterorientierten Zeichensatz verwendet. Somit erzeugen Sie auch einen Textstring mit Xor. Hierbei ist es allerdings notwendig, alle Zeichen einzeln, jeweils mit Xor-Anweisungen versehen und an der selbst errechneten Position auszugeben. Das hört sich etwas kompliziert und zeitraubend an, ist es aber nicht, und Turbo Pascal erledigt auch dies genügend schnell. Wenn Ihr Programm Grafiken erzeugt, die Sie frei beschriften möchten, können Sie nun Schriftzüge im Klartext mit der Maus zunächst probeweise positionieren und durch Drücken des Mausknopfes an der richtigen Stelle installieren oder wieder verwerfen. Auf diese Weise läßt sich auch jedes Zeichen oder jede Zeichenkombination als Cursor verwenden. Das Programm »xordemo.pas« (Listing)

demonstriert die Funktion des Xor-Put. Sie können hier mit optischen Hilfsmitteln Linien und Rechtecke zeichnen sowie Beschriftungen erarbeiten, zusätzlich kann zwischen drei verschiedenen Grafikzeigern gewählt werden. Zum Zeichnen benutzen Sie die Maus. Denken Sie vor dem Start daran, daß eine Maus angeschlossen und der Maustreiber geladen sein muß. Beim Abtippen ist zu beachten, daß Sie in Zeile 11 den Pfad, in dem sich Ihr BGI-Treiber befindet, angeben. Für professionelle Anwendungen empfiehlt es sich, die BGI-Treiber den unterstützten Grafikkarten mit dem im Lieferumfang des Compilers enthaltenen Programm »bin2obj« zusammenzufassen und zum Hauptprogramm hinzu zu linken.

(Karl-W. Schubotz/hs)

```
[Ø94]    1: { Programm: xordemo.pas
[7Z5]    2:   Funktion: demonstriert verknüpfte
[FE2]    3:             Grafikausgaben
[UY5]    4:   Sprache:  Turbo Pascal ab Version 4.Ø
[T44]    5:   Autor:    Karl-W. Schubotz
[P35]    6:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    7:
[A93]    8: USES Graph,Dos,Crt;
[ØØØ]    9:
[STØ]   1Ø: TYPE
[BK6]   11:   cursertype=  (fadenkreuz,kreuz,stift);
[1T5]   12:   zeichenhilfetype= (linie,rechteck,
[MH1]   13:                      schrift);
[6XØ]   14: CONST
[5D1]   15:   bgipath:
[HH4]   16:   STRING     ={ ..Suchweg für
[B63]   17:               EGAVGA.BGI-Treiber }
[UZ4]   18:   textzeichenabstand : BYTE = 8;
[RZ3]   19:   texthoehe          : BYTE = 1;
[262]   2Ø:   bewegen    = Ø;
[KD2]   21:   einblenden = 1;
[5F2]   22:   ausblenden = 2;
[9W2]   23:   xorstyle   = 4369Ø;
[TW1]   24:   zeile      = 1;
[RNØ]   25: VAR
[7B3]   26:   reg           :REGISTERS;
[VV1]   27:   stringxor,
[V32]   28:   textcurser,
[CN3]   29:   usertext      : STRING;
[CK4]   3Ø:   anker,mauspos : pointtype;
[EC4]   31:   aktcurs       : cursertype;
[WR1]   32:   aktzhilfe,
[YS4]   33:   obj           : zeichenhilfetype;
[R63]   34:   d,m,i         : INTEGER;
[563]   35:   xormodus      : BYTE;
[NRØ]   36:   key,
[NU2]   37:   funkey        : CHAR;
[G43]   38:   esc           : BOOLEAN;
[ØØØ]   39:
[4K6]   4Ø: PROCEDURE linexor(x1,y1,x2,y2:INTEGER);
[ØØØ]   41:
[MUØ]   42: BEGIN
[5F2]   43:   Port[$3CE]:= 3;
[ØK2]   44:   Port[$3CF]:= $18;
[6S2]   45:   Line(x1,y1,x2,y2);
[ZNØ]   46: END;
[ØØØ]   47:
[5Y5]   48: PROCEDURE rectanglexor(x1,y1,x2,y2:
[NK1]   49:                        INTEGER);
[ØØØ]   5Ø:
[MUØ]   51: BEGIN
[5F2]   52:   Port[$3CE]:= 3;
[ØK2]   53:   Port[$3CF]:= $18;
[4S2]   54:   Line(x1,y1,x1,y2);
[5F2]   55:   Port[$3CE]:= 3;
[ØK2]   56:   Port[$3CF]:= $18;
[9S2]   57:   Line(x1,y2,x2,y2);
[5F2]   58:   Port[$3CE]:= 3;
[ØK2]   59:   Port[$3CF]:= $18;
[CS2]   6Ø:   Line(x2,y2,x2,y1);
[5F2]   61:   Port[$3CE]:= 3;
[ØK2]   62:   Port[$3CF]:= $18;
[7S2]   63:   Line(x2,y1,x1,y1);
[ZNØ]   64: END;
[ØØØ]   65:
[8K5]   66: PROCEDURE textxor(x,y:INTEGER);
[ØØØ]   67:
[3Z2]   68: VAR   a,l,z : BYTE;
[ØØØ]   69:
[MUØ]   7Ø: BEGIN
[NS5]   71:   a:= texthoehe*textzeichenabstand;
[AR3]   72:   l:= Length(stringxor);
[7WØ]   73:   z:= Ø;
[P31]   74:   REPEAT
[B61]   75:     Inc(z);
[5F2]   76:     Port[$3CE]:= 3;
[ØK2]   77:     Port[$3CF]:= $18;
[1T5]   78:     OutTextXY(x-3,y-4,stringxor[z]);
[7S1]   79:     IF z=l THEN
[ØWØ]   8Ø:       Exit;
[271]   81:     x:= x+a;
[VH1]   82:   UNTIL 1=2;
[ZNØ]   83: END;
[ØØØ]   84:
[PV4]   85: FUNCTION maustaste :BOOLEAN;
[ØØØ]   86:
[MUØ]   87: BEGIN
[833]   88:   maustaste:= FALSE;
[RT1]   89:   reg.ax:= 5;
[B12]   9Ø:   Intr(51,reg);
[VD2]   91:   IF reg.bx>Ø THEN
[8Ø3]   92:     maustaste:= TRUE;
[ZNØ]   93: END;
[ØØØ]   94:
[NY2]   95: PROCEDURE curser;
[ØØØ]   96:
[MUØ]   97: BEGIN
[U23]   98:   IF (aktcurs=stift)
[AZ3]   99:     OR (aktcurs=kreuz) THEN
[MUØ]  1ØØ:     BEGIN
[6N5]  1Ø1:       textxor(mauspos.x,mauspos.y+1);
[F74]  1Ø2:       IF xormodus>bewegen THEN
[ØWØ]  1Ø3:         Exit;
[M34]  1Ø4:       textxor(reg.cx,reg.dx+1);
[ØWØ]  1Ø5:       Exit;
[ZNØ]  1Ø6:     END;
[146]  1Ø7:   linexor(Ø,mauspos.y,639,mauspos.y);
[CY6]  1Ø8:   linexor(mauspos.x,Ø,mauspos.x,GetMaxY);
[F74]  1Ø9:   IF xormodus>bewegen THEN
[ØWØ]  11Ø:     Exit;
[XN4]  111:   linexor(Ø,reg.dx,639,reg.dx);
[9S5]  112:   linexor(reg.cx,Ø,reg.cx,GetMaxY);
[ZNØ]  113: END;
[ØØØ]  114:
[MZ2]  115: PROCEDURE zhilfe;
[ØØØ]  116:
[MUØ]  117: BEGIN
[994]  118:   IF aktzhilfe=schrift THEN
[MUØ]  119:     BEGIN
[HA5]  12Ø:       textxor(mauspos.x,mauspos.y);
[F74]  121:       IF xormodus>bewegen THEN
[ØWØ]  122:         Exit;
[RZ3]  123:       textxor(reg.cx,reg.dx);
[ØWØ]  124:       Exit;
[ZNØ]  125:     END;
[PY3]  126:   IF aktzhilfe=linie THEN
[MUØ]  127:     BEGIN
[E36]  128:       linexor(anker.x,anker.y,mauspos.x,
[412]  129:               mauspos.y);
[F74]  13Ø:       IF xormodus>bewegen THEN
[ØWØ]  131:         Exit;
[AH5]  132:       linexor(anker.x,anker.y,reg.cx,
[C91]  133:               reg.dx);
[ØWØ]  134:       Exit;
[ZNØ]  135:     END;
[DR6]  136:   rectanglexor(anker.x,anker.y,mauspos.x,
[412]  137:                mauspos.y);
[F74]  138:   IF xormodus>bewegen THEN
[ØWØ]  139:     Exit;
[P36]  14Ø:   rectanglexor(anker.x,anker.y,reg.cx,
[C91]  141:                reg.dx);
[ZNØ]  142: END;
[ØØØ]  143:
[NY2]  144: PROCEDURE screen;
[ØØØ]  145:
[MUØ]  146: BEGIN
[322]  147:   ClearDevice;
[HT1]  148:   reg.ax:= 3;
[B12]  149:   Intr(51,reg);
[MD3]  15Ø:   mauspos.x:= reg.cx;
[SD3]  151:   mauspos.y:= reg.dx;
[MZ4]  152:   textcurser:= CHAR(17)+CHAR(24Ø) +
[XC1]  153:                CHAR(24Ø);
[GB4]  154:   stringxor:= textcurser;
[5W2]  155:   aktcurs:= stift;
[933]  156:   aktzhilfe:= linie;
[ZY1]  157:   obj:= linie;
[E32]  158:   SetColor(15);
[353]  159:   SetLineStyle(Ø,Ø,Ø);
[DU5]  16Ø:   Rectangle(5,18,GetMaxX-5,GetMaxY-14);
[1R4]  161:   SetLineStyle(4,xorstyle,Ø);
[U22]  162:   SetColor(1Ø);
[TS3]  163:   xormodus:= einblenden;
[CC1]  164:   curser;
[UB3]  165:   xormodus:= bewegen;
[EK5]  166:   usertext:= 'F1-Curser   F2-Objekt   '+
[XG4]  167:              'F3-Clear  ESC-Löschen/Ende';
[WU5]  168:   OutTextXY(1Ø,GetMaxY-12,usertext);
[582]  169:   usertext:= '';
[HN2]  17Ø:   GotoXY(3,zeile);
[YY3]  171:   Write('Zeichne Linie    ');
[ZNØ]  172: END;
[ØØØ]  173:
[C53]  174: PROCEDURE zeichne;
[ØØØ]  175:
```

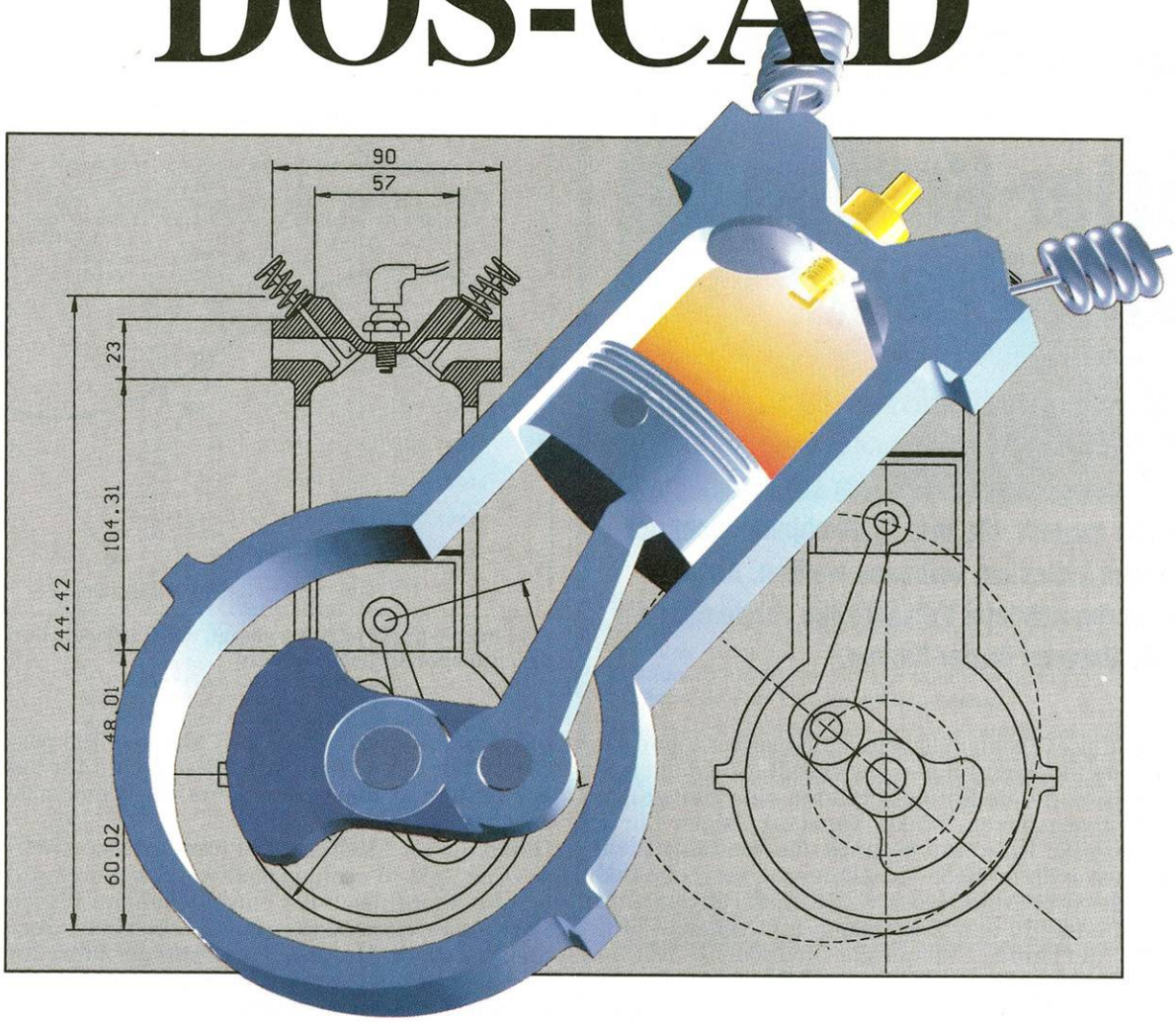
Listing. »xordemo.pas« veranschaulicht die Verwendung der schnellen Xor-Funktion

```
[5D1]  176: LABEL 1,2;
[ØØØ]  177:
[644]  178: VAR   hilfspunkte : BYTE;
[ØØØ]  179:
[ZX3]  18Ø: PROCEDURE setzeobjekt;
[ØØØ]  181:
[MUØ]  182: BEGIN
[E32]  183:   SetColor(15);
[353]  184:   SetLineStyle(Ø,Ø,Ø);
[RR2]  185:   IF obj=linie THEN
[BY5]  186:     Line(anker.x,anker.y,reg.cx,reg.dx);
[393]  187:   IF obj=rechteck THEN
[EE5]  188:     Rectangle(anker.x,anker.y,reg.cx,
[C91]  189:               reg.dx);
[Y53]  19Ø:   IF obj=schrift THEN
[MUØ]  191:     BEGIN
[HH4]  192:       OutTextXY(reg.cx-3,reg.dx-4,
[RT1]  193:                 usertext);
[ZNØ]  194:     END;
[U22]  195:   SetColor(1Ø);
[1R4]  196:   SetLineStyle(4,xorstyle,Ø);
[ZNØ]  197: END;
[ØØØ]  198:
[TC4]  199: PROCEDURE curserwechseln;
[ØØØ]  2ØØ:
[MUØ]  2Ø1: BEGIN
[WT3]  2Ø2:   xormodus:= ausblenden;
[CC1]  2Ø3:   curser;
[8H3]  2Ø4:   IF aktcurs=stift THEN
[MUØ]  2Ø5:     BEGIN
[ØP3]  2Ø6:       aktcurs:= fadenkreuz;
[ØWØ]  2Ø7:       Exit;
[ZNØ]  2Ø8:     END;
[MD4]  2Ø9:   IF aktcurs=fadenkreuz THEN
[MUØ]  21Ø:     BEGIN
[8U3]  211:       textcurser:= CHAR(197);
[GB4]  212:       stringxor:= textcurser;
[ØX2]  213:       aktcurs:= kreuz;
[ØWØ]  214:       Exit;
[ZNØ]  215:     END;
[KK3]  216:   IF aktcurs=kreuz THEN
[MUØ]  217:     BEGIN
[MZ4]  218:       textcurser:= CHAR(17)+CHAR(24Ø) +
[XC1]  219:                    CHAR(24Ø);
[GB4]  22Ø:       stringxor:= textcurser;
[5W2]  221:       aktcurs:= stift;
[ZNØ]  222:     END;
[ZNØ]  223: END;
[ØØØ]  224:
[MD3]  225: PROCEDURE clearkey;
[ØØØ]  226:
[MUØ]  227: BEGIN
[UW1]  228:   funkey:= ' ';
[WY1]  229:   esc:= FALSE;
[ZNØ]  23Ø: END;
[ØØØ]  231:
[XD3]  232: PROCEDURE keyboard;
[ØØØ]  233:
[5Ø1]  234: BEGIN;
[CC2]  235:   key:= ReadKey;
[FA2]  236:   IF key=#27 THEN
[AX1]  237:     esc:= TRUE;
[752]  238:   IF key=#Ø THEN
[9T2]  239:     funkey:= ReadKey
[ZRØ]  24Ø:   ELSE
[ØWØ]  241:     Exit;
[HM3]  242:   IF funkey=CHAR(59) THEN
[MUØ]  243:     BEGIN
[8G3]  244:       IF hilfspunkte=1 THEN
[Y53]  245:         IF obj=schrift THEN
[ØWØ]  246:           Exit;
[YW2]  247:       curserwechseln;
[TS3]  248:       xormodus:= einblenden;
[CC1]  249:       curser;
[UB3]  25Ø:       xormodus:= bewegen;
[ØWØ]  251:       Exit;
[ZNØ]  252:     END;
[8G3]  253:   IF hilfspunkte=1 THEN
[ØWØ]  254:     Exit;
[ØM3]  255:   IF funkey=CHAR(6Ø) THEN
[MUØ]  256:     BEGIN
[RR2]  257:       IF obj=linie THEN
[MUØ]  258:         BEGIN
[LL2]  259:           obj:= rechteck;
[1S2]  26Ø:           aktzhilfe:= obj;
[HN2]  261:           GotoXY(3,zeile);
[594]  262:           Write('Zeichne Rechteck ');
[ØWØ]  263:           Exit;
[ZNØ]  264:         END;
[Y53]  265:       IF obj=schrift THEN
[MUØ]  266:         BEGIN
[ZY1]  267:           obj:= linie;
[1S2]  268:           aktzhilfe:= obj;
[HN2]  269:           GotoXY(3,zeile);
[YY3]  27Ø:           Write('Zeichne Linie    ');
[ØWØ]  271:           Exit;
[ZNØ]  272:         END;
[SB2]  273:       obj:= schrift;
```

```
[1S2]  274:       aktzhilfe:= obj;
[HN2]  275:       GotoXY(3,zeile);
[4V3]  276:       Write('Setze Text       ');
[ZNØ]  277:     END;
[2M3]  278:   IF funkey=CHAR(61) THEN
[WA1]  279:     screen;
[ZNØ]  28Ø: END;
[ØØØ]  281:
[MUØ]  282: BEGIN
[EP2]  283:   hilfspunkte:= Ø;
[G8Ø]  284: 1:
[P31]  285:   REPEAT
[7H1]  286:     clearkey;
[Y23]  287:     IF KeyPressed THEN
[EL1]  288:       keyboard;
[HT1]  289:     reg.ax:= 3;
[B12]  29Ø:     Intr(51,reg);
[VN3]  291:     IF (reg.cx<>mauspos.x)
[UK4]  292:       OR (reg.dx<>mauspos.y) THEN
[MUØ]  293:       BEGIN
[3Z2]  294:         IF (hilfspunkte=Ø)
[PL3]  295:           OR (obj<>schrift) THEN
[CC1]  296:           curser;
[8G3]  297:         IF hilfspunkte=1 THEN
[3A1]  298:           zhilfe;
[MD3]  299:         mauspos.x:= reg.cx;
[SD3]  3ØØ:         mauspos.y:= reg.dx;
[ZNØ]  3Ø1:       END;
[S24]  3Ø2:   UNTIL maustaste OR esc;
[DX4]  3Ø3:   IF esc AND (hilfspunkte=1) THEN
[W61]  3Ø4:     GOTO 2;
[3S1]  3Ø5:   IF esc THEN
[ØWØ]  3Ø6:     Exit;
[7G3]  3Ø7:   IF hilfspunkte=Ø THEN
[MUØ]  3Ø8:     BEGIN
[YR2]  3Ø9:       anker.x:= reg.cx;
[6S2]  31Ø:       anker.y:= reg.dx;
[KP2]  311:       hilfspunkte:= 1;
[Y53]  312:       IF obj=schrift THEN
[MUØ]  313:         BEGIN
[WT3]  314:           xormodus:= ausblenden;
[CC1]  315:           curser;
[HN2]  316:           GotoXY(3,zeile);
[174]  317:           Write('Texteingabe: ');
[8V2]  318:           FOR i:= 16 TO 79 DO
[FM1]  319:             Write(' ');
[CV2]  32Ø:           GotoXY(16,zeile);
[4Y2]  321:           ReadLn(usertext);
[HN2]  322:           GotoXY(3,zeile);
[UP3]  323:           Write('Setze Text       ');
[8V2]  324:           FOR i:= 16 TO 79 DO
[FM1]  325:             Write(' ');
[3V3]  326:           stringxor:= usertext;
[ZNØ]  327:         END;
[TS3]  328:       xormodus:= einblenden;
[3A1]  329:       zhilfe;
[UB3]  33Ø:       xormodus:= bewegen;
[T61]  331:       GOTO 1;
[ZNØ]  332:     END;
[K8Ø]  333: 2:
[EP2]  334:   hilfspunkte:= Ø;
[WT3]  335:   xormodus:= ausblenden;
[UA3]  336:   IF obj<>schrift THEN
[CC1]  337:     curser;
[3A1]  338:   zhilfe;
[BB2]  339:   IF NOT esc THEN
[Y62]  34Ø:     setzeobjekt;
[GB4]  341:   stringxor:= textcurser;
[TS3]  342:   xormodus:= einblenden;
[CC1]  343:   curser;
[UB3]  344:   xormodus:= bewegen;
[T61]  345:   GOTO 1;
[ZNØ]  346: END;
[ØØØ]  347:
[MUØ]  348: BEGIN
[VUØ]  349:   d:= Ø;
[FVØ]  35Ø:   m:= 1;
[DB1]  351:   ClrScr;
[H92]  352:   TextColor(11);
[EU3]  353:   InitGraph(d,m,bgipath);
[8T1]  354:   reg.ax:= Ø;
[B12]  355:   Intr(51,reg);
[XT1]  356:   reg.ax:= 7;
[CT1]  357:   reg.cx:= Ø;
[UV2]  358:   reg.dx:= GetMaxX;
[B12]  359:   Intr(51,reg);
[ØU1]  36Ø:   reg.ax:= 8;
[CT1]  361:   reg.cx:= Ø;
[VV2]  362:   reg.dx:= GetMaxY;
[B12]  363:   Intr(51,reg);
[LF3]  364:   directvideo:= FALSE;
[PA2]  365:   SetBkColor(1);
[WA1]  366:   screen;
[LD1]  367:   zeichne;
[4V2]  368:   RestoreCrtMode;
[KNØ]  369: END.
```

Listing. »xordemo.pas« veranschaulicht die Verwendung der schnellen Xor-Funktion

DOS-CAD



Computerunterstütztes Konstruieren und Zeichnen

Ob Sie nun einen Schaltplan, einen Grundriß für eine Wohnungseinrichtung oder ein Strukturdiagramm entwerfen wollen, mit **DOS-CAD** nutzen Sie moderne CAD-Arbeitsmethoden und -techniken.

Die vielfältigen Funktionen umfassen das Zeichnen und Konstruieren von Polygonen, Rechtecken, Kreisen, Kreisbögen, Ellipsen, Doppellinien und vielem mehr. Komfortable Beschriftungsvarianten mit verschiedenen Zeichensätzen stehen Ihnen ebenso zur Verfügung wie alle wichtigen Funktionen zum Bemaßen der Zeichnung.

DOS-CAD arbeitet mit einer virtuellen Speicherverwaltung. Deshalb besteht keine Gefahr, daß Sie Ihre Zeichnung wegen zu geringer Speicherkapazität nicht vollenden können.

DOS-CAD ist ein leistungsfähiges und dennoch leicht zu erlernendes Programm: Die vielfältigen konstruktiven Möglichkeiten werden durch eine anwenderfreundliche und klar strukturierte Benutzeroberfläche überzeugend unterstützt.

Neben den Online-Hilfstexten bietet Ihnen das umfangreiche und leicht verständliche Handbuch Auskunft bei Anwendungsfragen wie zum Beispiel beim Fangen von Tangential- und Brennpunkten von Ellipsen.

Auch die Datenübergabe an andere Programme ist kein Problem. **DOS-CAD** gibt Konstruktionen im HPGL-, Postscript- oder DXF-Format aus.

DOS-CAD unterstützt verschiedenste Peripheriegeräte: Alle üblichen Grafikkarten , Mäuse und Grafiktabletts sowie Plotter, Matrix- oder Laserdrucker für die Zeichenausgabe.

Systemvoraussetzung:

- PC oder kompatibler mit mindestens 640 KByte RAM
- MS DOS ab Version 3.0
- zwei Diskettenlaufwerke
- Hercules-, CGA-, EGA- oder VGA-Grafikkarte
- Maus

DOS-CAD **DM 199,—***

Das leistungsfähige Konstruktionsprogramm

Neu im Programm:

Lightning 3D, Version 1.5 **DM 199,—***

Die faszinierende Welt der Grafik

* Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland 4,— DM bzw. für das Ausland 6,— DM Porto und Verpackung.

Bitte benutzen Sie die Bestellkarte.

DOS SOFTWARE

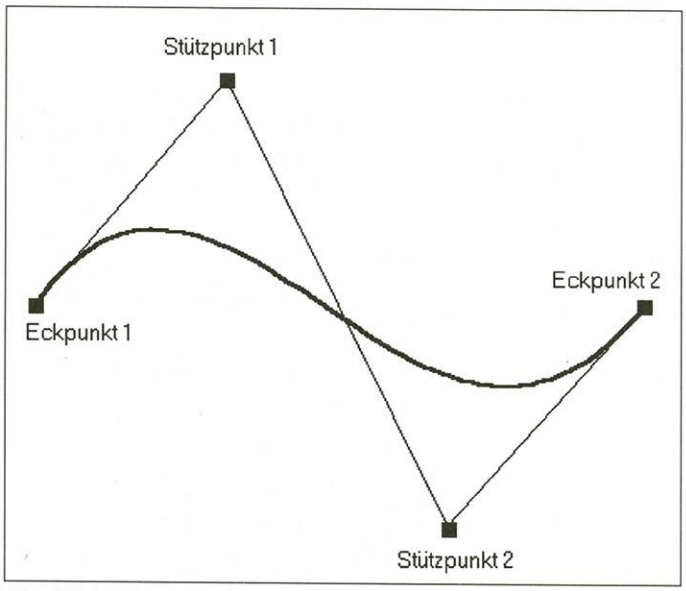
DMV-Verlag · Postfach 250 · 3440 Eschwege

DMV Daten- und Medienverlag

Turbo Pascal ab Version 5.0

Bézier-Kurven in Turbo Pascal

Wie zeichne ich eine sinusförmige Kurve auf den Bildschirm? Wie verbinde ich drei oder mehr Punkte? Diese und ähnliche Fragen haben sich schon viele Programmierer gestellt, die Ihre Grafikobjekte nicht nur aus kantigen Vielecken aufbauen wollten. Anstelle mühsam vorberechneter Polygonzüge verwenden wir die problemloseren Bézier-Kurven.



Eine Bézier-Kurve ist, kurz gesagt, eine Linie, die durch zwei oder drei Stützpunkte gebogen wird

Genau für dieses Problem haben Mathematiker schon früh entsprechende Algorithmen entwickelt. So erhalten Sie beispielsweise mit dem Spline-Verfahren eine glatte Kurve, welche die angegebenen Punkte annähert. Bézier-Kurven sind ein spezieller Fall dieser Splines und werden auch als B-Splines bezeichnet. Dabei bestimmen zwei Eckpunkte und zwei oder drei Stützpunkte die Form der Kurve. Diese beginnt und endet in den Eckpunkten und versucht in ihrem Verlauf, die beiden Stützpunkte anzunähern. Ein Beispiel einer Bézier-Kurve finden Sie im Bild.

Um die Rechenzeit so gering wie möglich zu halten, bedient man sich bei der Bézier-Kurven-Berechnung eines Tricks: Statt jeden einzelnen Punkt der Kurve zu berechnen, ist es sinnvoller, nur einige Punkte der Kurve zu beachten.

Auch Adobe erkannte die Vorteile der Bézier-Kurven für Postscript – bei wenig Rechenaufwand sehr glatte Kurven – und baute ihre Postscript-Zeichensätze darauf auf. Mit Bézier-Kurven ist es ein Kinderspiel, die Zeichen beliebig zu vergrößern und zu drehen, ohne Abstriche in der Schriftqualität hinzunehmen. Das Programm muß ja nur die Eck- und Stützpunkte skalieren beziehungsweise drehen, anstatt große Bitfelder zu bearbeiten. Mit »bspline.pas« (Listing 1) stellen wir Ihnen eine Turbo-Pascal-Unit für Bézier-Kurven zur Verfügung, die im Programm »bspldemo.pas« (Listing 2) anschaulich gemacht wird.

Durch die Unit stehen drei Prozeduren zur Wahl:

- »CalcSpl2« berechnet aus zwei Eck- und zwei Stützpunkten die Koordinaten der entsprechenden Bézier-Kurve. Die Kurve ist dabei in 32 (Konstante LPS, »Lines Per Spline«) Linien unterteilt. Der erste Parameter der Prozedur ist ein initialisiertes Punkt-Array vom Typ »PointArray«, der zweite Parameter ist ein erweitertes Punkt-Array vom Typ »BSpline-Array«. Nach Aufruf der Prozedur stehen die BSpline-Koordinaten im BSpline-Array.
- »CalcSpl3« entspricht »CalcSpl2«, jedoch verarbeitet diese Prozedur drei Stützpunkte.
- »DrawSpl« gibt schließlich die Bézier-Kurve aus. Dazu übergeben Sie als einzigen Parameter ein bereits berechnetes BSpline-Array. Die Kurve entsteht durch 32 Linien, die wiederum über den BGI-Befehl »LineTo« gezeichnet werden. Aus diesem Grund müssen Sie unbedingt das BGI vorher initialisieren.

(Martin Backschat/hs)

```
[374]    1: { Programm: bspline.pas
[HD5]    2:   Funktion: Unit zum Zeichnen von
[D82]    3:    Bezierkurven
[EK3]    4:   Sprache: Turbo Pascal
[E14]    5:   Autor: Martin Backschat
[P35]    6:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    7:
[M82]    8: UNIT bspline;
[ØØØ]    9:
[SK1]   1Ø: INTERFACE
[ØØØ]   11:
[CØ2]   12: USES Graph;
[ØØØ]   13:
[6XØ]   14: CONST
[2A1]   15:   lps = 32;
[STØ]   16: TYPE
[TA2]   17:   point = RECORD
[B92]   18:             x,y : INTEGER;
[ZNØ]   19:           END;
[7L6]   2Ø:   bsplinearray = ARRAY [Ø..lps] OF point;
[686]   21:   pointarray   = ARRAY [Ø..4]   OF point;
[ØØØ]   22:
[786]   23: PROCEDURE drawspl(sa : bsplinearray);
[L33]   24: PROCEDURE calcspl2
[NN6]   25:   (pa : pointarray; VAR sa : bsplinearray);
[P33]   26: PROCEDURE calcspl3
[NN6]   27:   (pa : pointarray; VAR sa : bsplinearray);
[ØØØ]   28:
[4L2]   29: IMPLEMENTATION
[ØØØ]   3Ø:
[VY5]   31: { Gibt die BSpline mit lineto(...) aus }
[786]   32: PROCEDURE drawspl(sa : bsplinearray);
```

```
[WN2]   33: VAR f : INTEGER;
[MUØ]   34: BEGIN
[TX3]   35:   MoveTo(sa[Ø].x,sa[Ø].y);
[WV2]   36:   FOR f := 1 TO lps DO
[N14]   37:     LineTo(sa[f].x,sa[f].y);
[ZNØ]   38: END;
[ØØØ]   39:
[ØØØ]   4Ø:
[EB7]   41: { Errechnet BSpline mit 2 Stuetzpunkten }
[L33]   42: PROCEDURE calcspl2
[NN6]   43:   (pa : pointarray; VAR sa : bsplinearray);
[K85]   44: VAR u,rØ,r1,r2,r3,r4,r5 : INTEGER;
[MUØ]   45: BEGIN
[463]   46:   sa[Ø].x := pa[Ø].x;
[B63]   47:   sa[Ø].y := pa[Ø].y;
[RV3]   48:   FOR u := 1 TO lps DO BEGIN
[232]   49:     rØ := pa[Ø].x+
[7S4]   5Ø:           ((u*(pa[1].x-pa[Ø].x)) DIV lps);
[732]   51:     r1 := pa[Ø].y+
[9S4]   52:           ((u*(pa[1].y-pa[Ø].y)) DIV lps);
[B32]   53:     r2 := pa[1].x+
[BS4]   54:           ((u*(pa[2].x-pa[1].x)) DIV lps);
[G32]   55:     r3 := pa[1].y+
[DS4]   56:           ((u*(pa[2].y-pa[1].y)) DIV lps);
[M32]   57:     r4 := pa[2].x+
[FS4]   58:           ((u*(pa[3].x-pa[2].x)) DIV lps);
[T32]   59:     r5 := pa[2].y+
[HS4]   6Ø:           ((u*(pa[3].y-pa[2].y)) DIV lps);
[694]   61:     rØ := rØ+((u*(r2-rØ)) DIV lps);
[F94]   62:     r1 := r1+((u*(r3-r1)) DIV lps);
[S94]   63:     r2 := r2+((u*(r4-r2)) DIV lps);
[1A4]   64:     r3 := r3+((u*(r5-r3)) DIV lps);
```

```
[U35]    65:      sa[u].x := rØ+((u*(r2-rØ)) DIV lps);
[345]    66:      sa[u].y := r1+((u*(r3-r1)) DIV lps);
[ZNØ]    67:    END;
[ZNØ]    68: END;
[ØØØ]    69:
[GB7]    7Ø: { Errechnet BSpline mit 3 Stuetzpunkten }
[P33]    71: PROCEDURE calcspl3
[NN6]    72:   (pa : pointarray; VAR sa : bsplinearray);
[U56]    73: VAR u,rØ,r1,r2,r3,r4,r5,r6,r7 : INTEGER;
[MUØ]    74: BEGIN
[463]    75:   sa[Ø].x := pa[Ø].x;
[B63]    76:   sa[Ø].y := pa[Ø].y;
[RV3]    77:   FOR u := 1 TO lps DO BEGIN
[232]    78:     rØ := pa[Ø].x+
[7S4]    79:           ((u*(pa[1].x-pa[Ø].x)) DIV lps);
[732]    8Ø:     r1 := pa[Ø].y+
[9S4]    81:           ((u*(pa[1].y-pa[Ø].y)) DIV lps);
[B32]    82:     r2 := pa[1].x+
[BS4]    83:           ((u*(pa[2].x-pa[1].x)) DIV lps);
[G32]    84:     r3 := pa[1].y+
[DS4]    85:           ((u*(pa[2].y-pa[1].y)) DIV lps);
[M32]    86:     r4 := pa[2].x+
[FS4]    87:           ((u*(pa[3].x-pa[2].x)) DIV lps);
[T32]    88:     r5 := pa[2].y+
[HS4]    89:           ((u*(pa[3].y-pa[2].y)) DIV lps);
[X32]    9Ø:     r6 := pa[3].x+
[LS4]    91:           ((u*(pa[4].x-pa[3].x)) DIV lps);
[242]    92:     r7 := pa[3].y+
[NS4]    93:           ((u*(pa[4].y-pa[3].y)) DIV lps);
[694]    94:     rØ := rØ+((u*(r2-rØ)) DIV lps);
[F94]    95:     r1 := r1+((u*(r3-r1)) DIV lps);
[S94]    96:     r2 := r2+((u*(r4-r2)) DIV lps);
[1A4]    97:     r3 := r3+((u*(r5-r3)) DIV lps);
[AA4]    98:     r4 := r4+((u*(r6-r4)) DIV lps);
[LA4]    99:     r5 := r5+((u*(r7-r5)) DIV lps);
[694]   1ØØ:     rØ := rØ+((u*(r2-rØ)) DIV lps);
[F94]   1Ø1:     r1 := r1+((u*(r3-r1)) DIV lps);
[S94]   1Ø2:     r2 := r2+((u*(r4-r2)) DIV lps);
[1A4]   1Ø3:     r3 := r3+((u*(r5-r3)) DIV lps);
[U35]   1Ø4:     sa[u].x := rØ+((u*(r2-rØ)) DIV lps);
[345]   1Ø5:     sa[u].y := r1+((u*(r3-r1)) DIV lps);
[ZNØ]   1Ø6:   END;
[ZNØ]   1Ø7: END;
[ØØØ]   1Ø8:
[KNØ]   1Ø9: END.
```

Listing 1. Mit der Turbo-Pascal-Unit »bspline.pas« errechnen und zeichnen Sie Bézier-Kurven

```
[L74]     1: { Programm: bspldemo.pas
[WK5]     2:   Funktion: Demo für Bezierkurven
[EK3]     3:   Sprache: Turbo Pascal
[E14]     4:   Autor: Martin Backschat
[P35]     5:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]     6:
[TM3]     7: PROGRAM bsplinedemo;
[ØØØ]     8:
[FC3]     9: USES bspline, Graph;
[ØØØ]    1Ø:
[RNØ]    11: VAR
[SH1]    12:   grdriver,
[E81]    13:   grmode,
[WØ3]    14:   errcode : INTEGER;
[Ø13]    15:   pa      : pointarray;
[EE3]    16:   sa      : bsplinearray;
[ØØØ]    17:
[MUØ]    18: BEGIN
[W13]    19:   grdriver :=detect;
[YK6]    2Ø:   InitGraph(grdriver,grmode,'a:\bspline');
[ØØØ]    21:
[K57]    22: { Bitte geben Sie anstelle 'a:\bspline'   }
[8S6]    23: { das Laufwerk und den Pfad auf Ihre      }
[S45]    24: { BGI-Treiber an.                         }
[ØØØ]    25:
[BV3]    26:   errcode:= GraphResult;
[LB4]    27:   IF errcode = grok THEN BEGIN
[MØ2]    28:     pa[Ø].x := Ø;
[AZ3]    29:     pa[1].x := GetMaxX DIV 3;
[893]    3Ø:     pa[2].x := pa[1].x*2;
[RA3]    31:     pa[3].x := GetMaxX;
[FZ3]    32:     pa[Ø].y := GetMaxY DIV 2;
[TØ2]    33:     pa[1].y := Ø;
[XA3]    34:     pa[2].y := GetMaxY;
[863]    35:     pa[3].y := pa[Ø].y;
[7D5]    36:     calcspl2(pa,sa);       {aus pa -> sa}
[FØ2]    37:     drawspl(sa);
[SZ2]    38:     WriteLn('Ready.');
[991]    39:     ReadLn;
[522]    4Ø:     CloseGraph;
[4MØ]    41:   END
[ZRØ]    42:   ELSE
[6E4]    43:     WriteLn('Graphics error:',
[914]    44:       GraphErrorMsg(errcode));
[KNØ]    45: END.
```

Listing 2. »bspldemo.pas« ist eine Demonstration für die Verwendung der BSpline-Unit

MaskEdit PLUS 5.0



Professioneller Maskengenerator

Nicht nur Profis sind begeistert

Auch viele Hobbyprogrammierer haben jetzt die Vorzüge eines professionellen Produktes entdeckt. Zudem ist durch die **kontextbezogene Hilfe** auf Tastendruck und die frei konfigurierbare Bedienung jeder in der Lage, sofort mit **MaskEdit Plus** konstruktiv zu arbeiten.

Zaubern auch Sie in Sekundenschnelle **komfortable Bildschirmmasken** für Menü- und Untermenüsysteme, Vorder- und Hintergrundmasken usw. auf den Monitor, und binden Sie danach den von **MaskEdit Plus** im gewählten Ausgabeformat erzeugten Quellcode einfach in Ihr Programm ein.

Geld sparen durch Modulsystem

Durch den modularen Aufbau von **MaskEdit Plus** können Sie gezielt und kostensparend einkaufen: Standard-Dateiausgabeformate wie ASCII, Binär, dBase und ANSI sind bereits im Grundpaket enthalten, durch **spezielle Maskentreiber** bedienen Sie Ihre bevorzugte Programmiersprache.

Professionelle Leistungen für jedermann

Neben acht Rahmentypen sind bis zu 16 Hintergrundfarben, Schattenwurf und Cursorkoordinaten darstellbar, und beim Rahmenzeichnen werden die **Kreuzungspunkte automatisch** angepaßt. Eine integrierte ASCII-Tabelle macht zusammen mit einem **universellen Maustreiber** gehörig Tempo.

Mit über 50 Funktionen können Sie auch **Teile einer Maske** kopieren, löschen, verschieben, zentrieren, färben oder füllen. Dabei wird von **MaskEdit Plus** der komplette IBM-Zeichensatz bis 255 Zeichen, die gleichzeitige Bearbeitung von zwei Masken und die Definition von Ein-/Ausgabefeldern unterstützt.

Programmier-Highlights der Maskentreiber in Kürze

Sämtliche Treiber werden als Sourcecode geliefert * Eigene Routinen können zur Kontrolle von Eingaben, Hilfetexten etc. problemlos in den Steuerungsablauf eingebunden werden * Integration von Datum und aktueller Uhrzeit und automatische Typüberprüfung bei der Eingabe, auch für den zusätzlichen Datentyp DATUM * Aufruf der generierten Masken durch einen einzigen Befehl und Steuerung der Ein-/Ausgabefelder durch den Maskentreiber * Wahlweise Ausschluß von Teilmengen des Zeichensatzes und automatische Eingabe in Großbuchstaben inklusive der Umlaute * Bei Änderungen von definierten Feldern kann die Kontrolle an das eigene Programm übergeben werden * Jedes Eingabefeld kann einzeln gefärbt werden, ist voll editierbar und läßt sowohl rechts- als auch linksbündige Ausgabe zu * Bewegung des Cursors innerhalb der Eingabemaske durch die Pfeil- und TAB-Tasten und Verlassen der Eingabemaske durch Funktionstasten-Betätigung.

Treiber:	**MaskEdit Plus**	
Turbo Pascal 3/4/5	mit einem Treiber	DM **99.-***
Quick C, Turbo C	Paketpreis mit zwei Treibern	DM **149.-***
TopSpeed Modula-2	Paketpreis mit drei Treibern	DM **199.-***
Turbo/Power Basic	Demonstrationsdiskette	DM **5.-**
Quick Basic ab V.4.0		

dazu passend empfehlen wir:

HYPERKEY 4.0, Tastaturbelegung DM **99.-***

* Unabhängig von der Anzahl der bestellten Produkte berechnen wir für das Inland 4.- DM, bzw. für das Ausland 6.- DM Porto und Verpackung.

Bitte benutzen Sie die Bestellkarte.

DMV-Verlag · Postfach 250 · 3440 Eschwege

DMV Daten- und Medienverlag

Turbo Pascal ab Version 4.0

Das BIOS auf Diskette

Alle Welt redet von Datensicherung. Nur ein »Programm« wird normalerweise nicht gesichert, das »Basic-Input-Output-System«, kurz BIOS genannt. Außerdem lassen sich in einem ausgelesenen BIOS Veränderungen vornehmen, zum Beispiel dann, wenn sich die benutzte Festplatte nicht in der vorhandenen Tabelle befindet, und eine freie Definition von Ihrem BIOS nicht zugelassen wird. Wir zeigen Ihnen, wie Sie die BIOS-Daten auf Diskette bekommen.

Das BIOS versteckt sich in der Regel in zwei elektrisch programmierbaren Festspeicherbausteinen, sogenannten EPROMs, und ist der Teil, der dem Computer erst Leben einhaucht. Ohne BIOS ist der Computer »dumm« und nicht einmal in der Lage, auch nur eine sinnvolle Funktion auszuführen.

Mit den Programmen »biosread.pas« und »biossort.pas« ist es möglich, das BIOS auszulesen und in der richtigen Form auf Diskette zu sichern. Bei einem 286er oder 386er sind meistens zwei EPROMs der Typen 27128 beziehungsweise 27256 vorhanden. Sie beinhalten 16 KByte (bei 27128) oder 32 KByte (bei 27256) an Daten. Welche Typen in Ihrem Computer Verwendung finden, entnehmen Sie bitte Ihrem Handbuch. Der Adreßbereich im Speicher beginnt bei F800:0000hex (bei 27128 EPROMs) und bei F000:0000hex (bei 27256 EPROMs). Mit dem Befehl »mem« (ab MS-DOS 4.0) kann man nun, abhängig von den verwendeten EPROM-Typen, die folgenden 32 oder 64 KByte einlesen und anschließend in die Datei »bios.dta« abspeichern. Die doppelte Kapazität der EPROMs deshalb, da ja immer zwei Stück vorhanden sind. Diese Aufgabe wird vom Programm »biosread.pas« (Listing 1) erfüllt. Bitte vergessen Sie beim Abtippen nicht, in Zeile 15 den richtigen EPROM-Typ einzugeben!

Für den Typ 27128:

```
biostyp = 128;
```

und für den Typ 27256:

```
biostyp = 256;
```

Da jedoch die Daten des BIOS auf zwei EPROMs verteilt sind, muß die Datei »bios.dta« aufgespaltet werden. Dabei ist es wichtig zu wissen, daß die Daten in den EPROMs nach einem bestimmten Verfahren abgelegt sind. Das erste Byte ist im Chip 1 (even) gespeichert, das zweite im Chip 2 (odd), das dritte wieder im Chip 1, etc. Falls Sie nun sagen, daß »eins« ja ungerade (odd) ist, so haben Sie zwar recht, aber da die Speicheradressen ab Null gezählt werden, ist die erste Adresse 6, und damit stimmt unsere Angabe wieder.

Das Aufspalten der Datei »bios.rom« erledigt das Programm »biossort.pas« (Listing 2). Es erzeugt aus der Datei »bios.dta« die beiden Dateien »bioseven.rom« und »biosodd.rom«.

Diese können nun mit einem EPROM-Brenngerät in die entsprechenden ICs geladen werden. Achten Sie aber auf die Zugriffszeit der verwendeten EPROMs. Sie sollte unter 150 ns liegen.

Sie können jedoch auch das BIOS mit einem Debugger betrachten und Ihre Veränderungen, wie den Eintrag eigener Festplattentypen, vornehmen. Beachten Sie aber bitte dann die Copyright-Bestimmungen des BIOS-Herstellers.

(Georg Pfaffenzeller/hs)

```
[T54]    1: { Programm: biosread.pas
[5P6]    2:   Funktion: speichert BIOS-Daten in Datei
[UY5]    3:   Sprache:  Turbo Pascal ab Version 4.Ø
[LS4]    4:   Autor:    Georg Pfaffenzeller
[P35]    5:   (c)1991 DMV Widuch GmbH & Co. KG }
[1YØ]    6:  {$N-}
[ØØØ]    7:
[5H1]    8:  USES Crt;
[ØØØ]    9:
[K36]   1Ø:  { Bitte unter biostyp das im Rechner
[VP4]   11:    befindliche BIOS eintragen:
[SØ3]   12:    128 : 128 KBIT EPROMs
[Ø13]   13:    z.B. 27C128 oder 27128
[YØ3]   14:    256 : 256 KBIT EPROMs
[1L3]   15:    z.B. 27C256 oder 27256     }
[ØØØ]   16:
[6XØ]   17:  CONST
[K72]   18:     biostyp = 128;
[ØØØ]   19:
[WM2]   2Ø:  VAR i  : LONGINT;
[RR2]   21:      bf : FILE OF BYTE;
[MUØ]   22: BEGIN
[XF1]   23:   WriteLn;
[ØG5]   24:   WriteLn('BiosRead (für AT-BIOS)');
[XF1]   25:   WriteLn;
[456]   26:   WriteLn('Im aktuellen Verzeichnis',
[7C3]   27:            ' wird ein Abbild des',
[EB4]   28:            ' RechnerBIOS in der Datei',
[ST1]   29:            ' bios.dta');
[KA3]   3Ø:   WriteLn('erzeugt.');
[XF1]   31:   WriteLn;
[3G3]   32:   Assign(bf,'bios.dta');
[Ø12]   33:   Rewrite(bf);
[163]   34:   IF biostyp = 256 THEN
[MUØ]   35:     BEGIN
[LØ3]   36:       FOR i := Ø TO 65535 DO
```

```
[MUØ]   37:         BEGIN
[ØH3]   38:           IF (i+1) MOD 1Ø24 = Ø THEN
[MUØ]   39:             BEGIN
[YY2]   4Ø:               GotoXY(1,WhereY);
[ZC3]   41:               Write( ((i+1) DIV 1Ø24):2,
[WN2]   42:                      '-tes KB (von 64)',
[PB2]   43:                      ' eingelesen');
[ZNØ]   44:             END;
[SN3]   45:           Write(bf,Mem[$FØØØ:i]);
[ZNØ]   46:         END;
[4MØ]   47:     END
[ZRØ]   48:   ELSE
[M53]   49:     IF biostyp = 128 THEN
[MUØ]   5Ø:       BEGIN
[TØ3]   51:         FOR i := Ø TO 32767 DO
[MUØ]   52:           BEGIN
[ØH3]   53:             IF (i+1) MOD 1Ø24 = Ø THEN
[MUØ]   54:               BEGIN
[YY2]   55:                 GotoXY(1,WhereY);
[ZC3]   56:                 Write( ((i+1) DIV 1Ø24):2,
[EN2]   57:                        '-tes KB (von 32)',
[PB2]   58:                        ' eingelesen');
[ZNØ]   59:               END;
[ØP3]   6Ø:             Write(bf,Mem[$F8ØØ:i]);
[ZNØ]   61:           END;
[ZNØ]   62:       END;
[CK1]   63:   Close(bf);
[XF1]   64:   WriteLn;
[SK1]   65:   WriteLn ;
[TS4]   66:   WriteLn('Auslesen beendet.');
[FH6]   67:   WriteLn('Die Nachbearbeitung kann mit',
[F24]   68:           ' dem Programm BIOSSORT',
[U22]   69:           ' erfolgen.');
[KNØ]   7Ø: END.
```

Listing 1. Das Programm »biosread.pas« speichert die BIOS-Daten in der Datei »bios.dta«

```
[D94]    1: { Programm: biossort.pas
[K66]    2:   Funktion: spaltet BIOS-Datei in zwei
[UY5]    3:   Sprache:  Turbo Pascal ab Version 4.Ø
[LS4]    4:   Autor:    Georg Pfaffenzeller
[P35]    5:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    6:
[1YØ]    7: {$N-}
[ØØØ]    8:
[5H1]    9: USES Crt;
[ØØØ]   1Ø:
[H14]   11: VAR even,Odd           : BYTE;
[DC5]   12:     evenf,oddf, infile : FILE OF BYTE;
[563]   13:     i                  : LONGINT;
[MUØ]   14: BEGIN
[XF1]   15:   WriteLn;
[2N6]   16:   WriteLn('BiosSort V1.Ø (für AT-BIOS)');
[XF1]   17:   WriteLn;
[Ø46]   18:   WriteLn('Die Datei bios.dta wird in',
[1U3]   19:           ' even- und odd-Anteil ',
[665]   2Ø:           'gespalten. Die entstehenden');
[XK6]   21:   WriteLn('Files bioseven.rom (Low) und',
[6U4]   22:           ' biosodd.rom (High) können',
[3V4]   23:           ' anschließend in EPROMs');
[ES6]   24:   WriteLn('der entsprechenden Typen (mit',
[VB4]   25:           ' Zugriffszeiten <= 15Øns)',
[G24]   26:           ' übertragen werden.');
[SK1]   27:   WriteLn ;
[3P4]   28:   Assign(evenf,'bioseven.rom');
[DG4]   29:   Assign(oddf,'biosodd.rom');
[674]   3Ø:   Assign(infile,'bios.dta');
```

```
[NB2]   31:   Reset(infile);
[CP2]   32:   Rewrite(evenf);
[ED2]   33:   Rewrite(oddf);
[NA6]   34: { Aufteilung in Even- und Oddanteil }
[5VØ]   35:   i:= Ø;
[NT3]   36:   WHILE NOT EoF(infile) DO
[MUØ]   37:     BEGIN
[4N3]   38:       Read(infile,even,Odd);
[H13]   39:       Write(evenf,even);
[AH2]   4Ø:       Write(oddf,Odd);
[791]   41:       i:= i + 2;
[X53]   42:       IF (i MOD 1Ø24) = Ø THEN
[MUØ]   43:         BEGIN
[YY2]   44:           GotoXY(1,WhereY);
[V84]   45:           Write((i DIV 1Ø24):2,'-tes KB',
[E62]   46:           ' gespalten');
[ZNØ]   47:         END;
[ZNØ]   48:     END;
[G42]   49:   Close(evenf);
[MX1]   5Ø:   Close(oddf);
[CA2]   51:   Close(infile);
[SK1]   52:   WriteLn ;
[SK1]   53:   WriteLn ;
[RT6]   54:   WriteLn('Sortierung abgeschlossen. Die',
[MP4]   55:           ' EPROMs können gebrannt ',
[BT1]   56:           'werden.');
[KNØ]   57: END.
```

Listing 2. »biossort.pas« spaltet das ausgelesene BIOS in die Dateien »biosodd.rom« und »bioseven.rom« auf

MASM ab Version 4.0/TASM ab Version 1.0

Batch-Dateien werden Maus-fähig

Warum die Maus nicht auch in Batch-Dateien benutzen, wenn man sich damit die Arbeit am PC etwas erleichtern kann? GEM, Windows und Geo Works Ensemble sowieso, Word und auch Word Perfect – immer mehr Programme kann man mit der Maus bedienen. Das kleine »Nagetier« erleichtert zahlreiche Operationen. Mit der Zeit gewöhnt man sich an diesen Eingabekomfort.

An diesem Punkt angelangt, ist es dann besonders lästig, wenn man auf Betriebssystemebene doch wieder auf die Tastatur zurückgreifen muß. Zwar kann man sich die Arbeit etwas erleichtern, wenn man sich einige kleine Menüs aus Batch-Dateien erstellt, um seine Lieblingsprogramme aufzurufen. Mit Nortons »ask« oder einem der vielen anderen Programme, die eine Abfrage in Stapeldateien ermöglichen (beim Betriebssystem hat man darauf keinen Wert gelegt), kann man dann einfach durch Tastendruck seine Textverarbeitung oder die Datenbank starten. Aber, wie ärgerlich: Man muß eben immer noch zur Tastatur greifen, obwohl man die Hand gerade auf der Maus hatte.

Aus dieser Überlegung heraus entstand »mausask.com« (Listing 1).

Nun können Sie auch den Namen des Programms »entschlüsseln«: Er setzt sich aus »maus« und »ask« zusammen. Ist kein Mausteiber geladen, verhält sich das Programm wie jedes andere Abfrageprogramm auch: Sie drücken eine der Tasten, die Sie in der Kommandozeile angegeben haben, und »mausask« liefert Ihnen brav einen Exitcode zurück, der mit »Errorlevel« in der Batch-Datei ausgewertet werden kann. Haben Sie jedoch eine Maus, und der Maustreiber ist geladen, so können Sie Ihr kleines Batch-Menü nun auch mit der Maus bedienen. Der Maus-Zeiger erscheint, und Sie können die richtige Menüzeile auswählen.
Wenn Sie Ihre Stapelmenüs auch bisher schon in mehreren Zeilen mit dem Echo-Befehl (oder »type«, »copy con« etc.) auf dem Bildschirm angezeigt haben, brauchen Sie an dem Menü sogar nichts zu ändern, außer vielleicht dem Einfügen eines Cls-Befehls am Anfang (damit das Menü immer in den gleichen Zeilen auf dem Bildschirm erscheint), und dem Ersetzen der alten Abfrage durch »mausask.com«.

Problemloser Mauskomfort in Batch-Dateien

Mausask erkennt eine angeschlossene Maus selbständig, so daß das Programm mit oder ohne Maus betrieben werden kann. Das Besondere aber ist die Möglichkeit, Batch-Dateien mit der Maus zu bedienen. Sollten Sie noch keine Maus besitzen, ist dies ja vielleicht eine Gelegenheit, über eine Anschaffung nachzudenken.
Die Kommandozeilen-Syntax von »mausask« ist gegenüber anderen Abfrageprogrammmen so erweitert, daß neben den »erlaubten« Tasten zusätzlich auch noch eine dazugehörende Bildschirmzeile angegeben werden muß. Bisher haben Sie also in Ihrem kleinen Startmenü beispielsweise angegeben:

```
echo (T)extverarbeitung
echo (D)atenbank
echo (Q)uit)
frage.com TDQ
if errorlevel ...
```

Zuvor haben Sie sich, wie schon angedeutet, mit dem Echo-Befehl einige Zeilen Informations-Text ausgeben lassen. Mit »mausask« würde der vorherige Programmaufruf dann etwa so aussehen:

```
mausask aTbDcQ
```

Erkennen Sie Ihre »Tasten« T, D und Q wieder? Vorher ist jetzt jeweils zusätzlich die Zeilennummer (als Kleinbuchstabe von »a« bis »y«) angegeben, in der dieser Menüpunkt

dargestellt wird. Dies ist die zusätzliche Information für die Maus. Die Information, für welche Bildschirmzeile welche »Taste« zurückgegeben werden soll, ist in der Kommandozeile enthalten. Mausask arbeitet also ohne »Tricks«. Das Programm ist weder resident, noch wird der Wert, der von der Maus zurückgegeben werden soll, direkt aus dem Bildschirmspeicher entnommen. Die Parameter in der Kommandozeile sind die entscheidenden Informationen für das Programm. Kleinbuchstaben für die Zeileninformation wurden gewählt, damit man für die Zeilennummern mit jeweils einem Zeichen auskommt und die Werte trotzdem leicht umrechenbar sind.
Ihr Stapelmenü könnte nun etwa folgendermaßen aussehen:

```
@echo off
cls
rem CLS nicht vergessen,
rem damit die Zeilennummern stimmen
echo \
echo \
echo (T)extverarbeitung starten
echo (D)atenbank laden
echo (Q)uit
mausask cTdDeQ
if errorlevel 3 goto ende
if errorlevel 2 goto text
if errorlevel 1 goto daten
:text
supertext.exe
goto ende
:daten
wowbase.exe
:ende
rem Ende der Stapeldatei
```

Nebenbei eine Information zum Errorlevel: Wenn Sie sich die Batch-Datei ansehen, werden Sie bemerken, daß die »Errorlevel« von groß nach klein abgefragt werden. Das muß leider so sein. Als »vereidigter alter Hase« wissen Sie sicher, daß sich DOS bei der Auswertung des Exitcodes den Spaß erlaubt, jeweils »größer oder gleich« abzufragen, was die Auswertung (nicht nur für Anfänger) nicht gerade erleichtert. Deshalb müssen immer zuerst die größten Werte geprüft werden.
Außerdem kann man erkennen, daß Mausask (vielleicht anders als Ihr Ask-Programm) nicht den ASCII-Wert der Taste als Exit-Code zurückliefert, sondern die Position des Zeichens in der Kommandozeile. Dabei werden Bildschirmzeile und zugehörige Taste zusammen als eine Stelle bewertet. Der erste Parameter (zwei Zeichen) liefert demnach den Errorlevel 1 zurück, das zweite Parameterpärchen den Exitcode 2, etc., was die Auswertung vereinfacht. Es wird keine ASCII-Tabelle mehr benötigt, um sich »auf die Schnelle« ein kleines Stapelmenü zu erstellen.

Mausask ist bis auf eine Ausnahme »sauber« programmiert, das heißt, alle Ein- und Ausgaben werden über Interrupt-Aufrufe bewerkstelligt. Deshalb ist auch ohne weiteres eine Umleitung zum »Nul-Device« möglich, um beispielsweise das Piepen zu unterdrücken, das bei einer falsch gedrückten Taste beziehungsweise beim Drücken der linken Maustaste auf einer »falschen« Bildschirmzeile ausgegeben wird. Die besagte Ausnahme ist eigentlich nur für einige ältere DOS-Versionen erforderlich, bei denen der Maus-Interrupt (Int 33h) beim Starten des Systems nicht mit einem Zeiger auf »iret« initialisiert wird. Wird dann nämlich, ohne vorherige Kontrolle, ob das überhaupt »erlaubt« ist, der Maus-Interrupt aufgerufen, stürzt der Computer ab. Deshalb prüft »mausask« am Anfang zunächst »zu Fuß«, ob an der Adresse 0000:00CE ein Zeiger eingetragen ist, der eventuell auf eine andere Adresse als 0000:0000 zeigt. Selbverständlich kann vor den Parametern auch ein Text in der Kommandozeile angegeben werden, der auf dem Bildschirm angezeigt werden soll. Dieser Text kann beliebige Zeichen enthalten, zum Beispiel auch Leerzeichen. Der Text muß in Anführungszeichen eingeschlossen und mit einem Komma abgeschlossen sein, also in der Form:

```
mausask "TextText",cTdDeQ
```

Bitte beachten Sie, daß die Bildschirmzeilen als Kleinbuchstaben von »a« bis »y« (von oben nach unten gezählt) angegeben werden. Bei den Tasten sind, bis auf die Sonderzeichen (dazu gehören in diesem Fall auch Ä, Ö etc.), Klein- und Großbuchstaben erlaubt. Alle Tasten, die Scancode erzeugen (Funktionstasten, Cursortasten etc.), können nicht ausgewertet werden.
Allerdings läßt sich mit einem Trick ein Standardwert schaffen, den Sie dann mit der Enter-Taste erreichen. Geben Sie einfach hinter dem letzten Parameter ein einziges Leerzeichen ein, dann ist der letzte Parameter der Standardwert, der durch [Enter] ausgewählt wird. Ohne Parameter aufgerufen, gibt das Programm eine kleine Hilfestellung aus. Obwohl das Programm sowohl mit der Hercules-Karte als auch mit einer VGA-Karte getestet wurde, könnten theoretisch Probleme bei der Berechnung der Bildschirmzeile für die Maus auftreten. In einem solchen Fall müßten Sie in Listing 2 (»mausask.asm«) für die Berechnung dieses Wertes einen entsprechenden Korrekturwert einfügen.
Abschließend noch ein Tip: Wenn Sie den Programmnamen von »mausask« auf drei Stellen verkürzen, also das Programm etwa »msa.com« nennen, hat das den Vorteil, daß die Texteingabe in der Stapeldatei genau an der gleichen Stelle steht, wie bei der Eingabe einer Echo-Zeile.

(Jochen Bleichroth/bj)

```
[CH5]    1: B8 ØØ ØC CD 21 8A 3E 8Ø ØØ 8Ø FF Ø2 72 18 B3 81
[PS5]    2: Ø2 DF 32 FF 8B FB BE 82 ØØ E8 AF ØØ 74 Ø3 4E EB
[HM5]    3: ØE AC 3C 2C 74 Ø9 BA Ø8 Ø2 BØ FF 5Ø E9 93 ØØ 89
[MP5]    4: 36 Ø6 Ø2 AD E8 AB ØØ 3B F7 72 F8 33 CØ 8E CØ 26
[8M5]    5: 39 Ø6 CE ØØ 74 Ø2 CD 33 3D FF FF 75 ØA C6 Ø6 Ø5
[ZD5]    6: Ø2 Ø1 B8 Ø1 ØØ CD 33 8B 36 Ø6 Ø2 B4 Ø1 CD 16 74
[7R5]    7: 16 B4 ØØ CD 16 8B D8 AD E8 88 ØØ 72 37 3B F7 72
[MG5]    8: F6 B4 Ø2 B2 Ø7 CD 21 8Ø 3E Ø5 Ø2 Ø1 75 D9 33 DB
[XK5]    9: B8 Ø5 ØØ CD 33 83 FB ØØ 74 CD 8B 36 Ø6 Ø2 AD 32
[4T5]   1Ø: E4 2C 61 B1 Ø3 D3 EØ 3B DØ 74 Ø6 3B F7 72 EF EB
[YL5]   11: DØ 8A 5C FF 2B 36 Ø6 Ø2 D1 EE 8B C6 5Ø 8Ø 3E Ø5
[WL5]   12: Ø2 Ø1 75 Ø5 B8 Ø2 ØØ CD 33 8A D3 B4 Ø2 CD 21 BA
[SN5]   13: 89 Ø2 B4 Ø9 CD 21 58 B4 4C CD 21 AC 3C 22 75 11
[ØT5]   14: AC 3B F7 74 ØC 3C 22 74 Ø8 8A DØ B4 Ø2 CD 21 EB
[BN5]   15: EF C3 8Ø FC 61 72 ØB 8Ø FC 7A 77 Ø6 8Ø EC 2Ø 88
[9R5]   16: 64 FF C3 3A E3 74 ØC 8A FB 8Ø EF 2Ø 3A E7 74 Ø3
[FD5]   17: F8 EB Ø1 F9 C3 ØØ ØØ ØØ 4D 41 55 53 41 53 4B 2Ø
[E85]   18: 76 31 2E 3Ø 3Ø 2C 2Ø 28 63 29 2Ø 31 39 39 31 2Ø
[SF5]   19: 44 4F 53 2Ø 49 6E 74 65 72 6E 61 74 69 6F 6E 61
[ZD5]   2Ø: 6C ØD ØA 2Ø 2Ø 2Ø 2Ø 41 75 74 6F 72 3A 2Ø 4A 6F
[8H5]   21: 63 68 65 6E 2Ø 42 6C 65 69 63 68 72 6F 74 68 ØD
[ME5]   22: ØA ØD ØA 41 75 66 72 75 66 3A 2Ø 6D 61 75 73 61
[PF5]   23: 73 6B 2Ø 5B 22 54 65 78 74 22 2C 5D 42 69 6C 64
[NG5]   24: 73 63 68 69 72 6D 7A 65 69 6C 65 28 61 2E 2E 79
[T34]   25: 29 54 61 73 74 65 2E 2E 2E ØD ØA 24
```

Listing1. »mausask.com« ist der ablauffähige Hexdump des Programms »mausask.asm«

```
[N34]    1: ; Programm: MausAsk.asm
[VV5]    2: ; Funktion: Abfrage in Batch-Dateien
[NV4]    3: ;           mit Tastatur und Maus.
[AC3]    4: ; Sprache:  Assembler
[5G4]    5: ; Autor:    Jochen Bleichroth
[FU4]    6: ; (c)1991 DMV Widuch GmbH & Co. KG
[ØØØ]    7:
[ØØØ]    8:
[G73]    9: CODE       SEGMENT BYTE
[TW3]   1Ø: ASSUME     CS:CODE,DS:CODE
[AG1]   11:            ORG     1ØØh
[ØØØ]   12:
[ØØØ]   13:
[5A3]   14: START:     mov     ax,ØcØØh
[DR5]   15:            ;Tastaturpuffer über DOS leeren
[N81]   16:            int     21h
[TN2]   17:            mov     bh,ds:[8Øh]
[CØ5]   18:            ;Parameterlänge feststellen
[BD1]   19:            cmp     bh,2
[TE4]   2Ø:            ;Sind Parameter angegeben?
[CN1]   21:            jb      USAGE
[ZR1]   22:            mov     bl,81h
[R54]   23:            ;Länge festhalten in di
```

```
[GN1]   24:             add   bl,bh
[LT1]   25:             xor   bh,bh
[PV1]   26:             mov   di,bx
[4U1]   27:             mov   si,82h
[SG2]   28:             call  TEST_TXT
[VB3]   29:             ;Ist Text angegeben?
[UD2]   3Ø:             je    TESTCOMMA
[L45]   31:             ;Wenn ja, folgt dann ein Komma?
[S51]   32:             dec   si
[FR2]   33:             jmp   short OKAY
[A44]   34:             ;Wenn kein Text, dann los!
[3U2]   35: TESTCOMMA:lodsb
[DK1]   36:             cmp   al,","
[C33]   37:             ;Ist es ein Komma?
[RD1]   38:             je    OKAY
[SX2]   39:             ;Wenn ja, dann los!
[CM5]   4Ø: USAGE:      mov   dx,offset USAGE_TXT
[AT3]   41:             ;Wenn nein, dann Ende ...
[5Y1]   42:             mov   al,Øffh
[Z33]   43:             ;... mit Exitcode 255
[F91]   44:             push  ax
[ZM2]   45:             jmp   TERMINATE
[ØØØ]   46:
[N84]   47: OKAY:       mov   ds:[PARAM],si
[6S5]   48:             ;Parameter-Start-Stelle speichern
[Y62]   49: UP:         lodsw
[GØ5]   5Ø:             ;'Tasten' in Großbuchstaben
[3Z1]   51:             call  UPCASE
[R62]   52:             ;umwandeln ...
[AU1]   53:             cmp   si,di
[Y11]   54:             jb    UP
[FN3]   55:             ;... bis Parameter-Ende
[ØØØ]   56:
[SH4]   57:             ;Bei manchen (älteren) DOS-
[ZM4]   58:             ;Versionen ist der Maus-Int
[LF3]   59:             ;noch nicht mit IRET
[XZ4]   6Ø:             ;initialisiert. Deshalb Test.
[WV1]   61:             xor   ax,ax
[1W1]   62:             mov   es,ax
[8Z2]   63:             cmp   es:[Øceh],ax
[M82]   64:             jz    NO_MOUSE
[V81]   65:             int   33h
[3Ø4]   66: NO_MOUSE:   cmp   ax,Øffffh
[XR5]   67:             ;Maus-Driver liefert FFFF zurück,
[2F3]   68:             ;wenn Maus vorhanden.
[L62]   69:             jnz   ASK_KEY
[774]   7Ø:             mov   ds:byte ptr [FLAG],1
[CN4]   71:            ;Wenn Maus, dann Flag setzen
[6F1]   72:             mov   ax,1
[V81]   73:             int   33h
[824]   74:             ;Maus-Zeiger anschalten
[ØØØ]   75:
[ØR4]   76: ASK_KEY:    mov   si,ds:[PARAM]
[ZZ3]   77:             ;Anfangs-Parameter laden
[PE1]   78:             mov   ah,1
[P81]   79:             int   16h
[XB4]   8Ø:             ;Ist eine Taste gedrückt?
[6H2]   81:             jz    ASK_MOUSE
[GX4]   82:             ;Nein, wenn Zero-Flag gesetzt!
[VP1]   83:             mov   ah,ØØh
[P81]   84:             int   16h
[7V3]   85:             ;Wenn ja, welche Taste?
[UV1]   86:             mov   bx,ax
[LU2]   87: TEST_KEY:   lodsw
[A12]   88:             call  COMPARE
[RA5]   89:             ;Taste mit Parameter vergleichen
[922]   9Ø:             jc    EXITCODE
[MV2]   91:             ;Wenn Taste okay,
[YE3]   92:             ;dann Exitcode setzen
[AU1]   93:             cmp   si,di
[D92]   94:             jb    TEST_KEY
[GH5]   95:             ;Mit allen Parametern vergleichen
[PH2]   96: BEEP:       mov   ah,2
[7M4]   97:             ;Falsche Taste, also Piepen
[SF1]   98:             mov   dl,7
[N81]   99:             int   21h
[ØØØ]   1ØØ:
[N36]   1Ø1: ASK_MOUSE:cmp   ds:byte ptr [FLAG],1
[B44]   1Ø2:            ;Nur wenn Maus aktiviert
[A34]   1Ø3:            ;in diesen Teil springen
[L62]   1Ø4:            jnz   ASK_KEY
[3W1]   1Ø5:            xor   bx,bx
[BH4]   1Ø6:            ;(Für Linkshänder: mov bx,1)
[KF1]   1Ø7:            mov   ax,5
[YA6]   1Ø8:            ;Abfrage: linke Maustaste gedrück
        t?
[V81]   1Ø9:            int   33h
[ND1]   11Ø:            cmp   bx,Ø
[RZ3]   111:            ;Wurde Taste gedrückt?
[FØ2]   112:            jz    ASK_KEY
[XV4]   113:            ;Wenn nein, dann alles nochmal
[ØØØ]   114:
[RD3]   115:            mov   si,ds:[PARAM]
[5Z5]   116:            ;Wenn ja, dann Parameter-Anfang l
        aden
[XZ2]   117: TEST_MOUS:lodsw
[CT1]   118:            xor   ah,ah
[SV2]   119:            ;Klein-Buchstaben
```

```
[173]   12Ø:            ;(=Bildschirmzeile)
[UY4]   121:            ;aus Kommandozeile auswerten
[3P1]   122:            sub   al,61h
[8F1]   123:            mov   cl,3
[R55]   124:            ;Buchstabe in 'Zeile' umwandeln
[VS1]   125:            shl   ax,cl
[GU1]   126:            cmp   dx,ax
[AT2]   127:            ;Richtige Zeile?
[V52]   128:            je    MOUSE_OK
[KT4]   129:            ;Wenn ja, dann also Maus okay!
[AU1]   13Ø:            cmp   si,di
[3M2]   131:            jb    TEST_MOUS
[SB5]   132:            ;Wiederholen bis Parameter-Ende
[AP2]   133:            jmp   short BEEP
[Ø24]   134:            ;Leider nein, also Piepen
[8G4]   135: MOUSE_OK:  mov   bl,ds:[si-1]
[WM5]   136:            ;Okay, also zugehörigen Tasten-
[9X1]   137:            ;code laden
[ØØØ]   138:
[LP4]   139: EXITCODE:  sub   si,ds:[PARAM]
[U45]   14Ø:            ;Aus Parameter-Stelle den Exit-
[MV2]   141:            ;code berechnen...
[6G1]   142:            shr   si,1
[GV1]   143:            mov   ax,si
[F91]   144:            push  ax
[VC3]   145:            ;... und in AL sichern
[N54]   146:            cmp   ds:byte ptr [FLAG],1
[WY4]   147:            ;Jetzt die Maus ausschalten...
[8E2]   148:            jnz   MOUSE_NO
[T73]   149:            ;...sofern vorhanden
[9F1]   15Ø:            mov   ax,2
[V81]   151:            int   33h
[NA3]   152: MOUSE_NO:  mov   dl,bl
[9U2]   153:            ;'Taste' ausgeben
[TE1]   154:            mov   ah,2
[N81]   155:            int   21h
[7W3]   156:            mov   dx,offset CR_TXT
[A94]   157:           ;Carriage-Return ausgeben
[A43]   158: TERMINATE:mov   ah,9
[N81]   159:            int   21h
[971]   16Ø:            pop   ax
[7S4]   161:            ;Gesicherten Exitcode laden
[LT1]   162:            mov   ah,4ch
[W25]   163:            ;... und damit Programm beenden
[N81]   164:            int   21h
[ØØØ]   165:
[1V2]   166: TEST_TXT:  lodsb
[TG4]   167:            ;Kommandozeile auf "Text" ...
[6Z3]   168:            ;... untersuchen, der von
[4D3]   169:            ;Anführungszeichen
[SF4]   17Ø:            ;eingeschlossen sein muß!
[TH1]   171:            cmp   al,'"'
[952]   172:            jne   TXT_END
[9V2]   173: NEXT_TXT:  lodsb
[AU1]   174:            cmp   si,di
[UZ1]   175:            je    TXT_END
[TH1]   176:            cmp   al,'"'
[X84]   177:            ;Text endet, wenn zweites
[SZ4]   178:            ;Anführungszeichen gefunden
[UZ1]   179:            je    TXT_END
[6U1]   18Ø:            mov   dl,al
[455]   181:            ;Wenn Text da, einzeln ausgeben
[TE1]   182:            mov   ah,2
[D85]   183:            ;Der Text kann auch Leerzeichen
[ØX1]   184:            ;enthalten!
[N81]   185:            int   21h
[ØN3]   186:            jmp   short NEXT_TXT
[K72]   187: TXT_END:   ret
[DA1]   188:            ;Fertig
[ØØØ]   189:
[R33]   19Ø: UPCASE:    cmp   ah,61h
[4F5]   191:            ;Ist die 'Taste' in der Kommando-
[YF5]   192:            ;zeile ein Kleinbuchstabe, dann i
        n
[B96]   193:            ;einen Großbuchstaben umwandeln..
        .
[4V1]   194:            jb    RETURN
[2S1]   195:            cmp   ah,7ah
[1V1]   196:            ja    RETURN
[VS5]   197:            ;So werden nachher zwei Vergleich
        e
[SP1]   198:            ; gespart !)
[FN1]   199:            sub   ah,2Øh
[3Ø3]   2ØØ:            mov   ds:[si-1],ah
[3Y3]   2Ø1:            ;...und in Kommandozeile
[B23]   2Ø2:            ;zurückschreiben
[342]   2Ø3: RETURN:    ret
[DA1]   2Ø4:            ;Fertig
[ØØØ]   2Ø5:
[U13]   2Ø6: COMPARE:   cmp   ah,bl
[Z16]   2Ø7:            ;Kommandozeile enthält nur Groß-
[K71]   2Ø8:            je    YES
[AX4]   2Ø9:            ;buchstaben, wenn also Tasten
[XT1]   21Ø:            mov   bh,bl
[575]   211:            ;nicht gleich, dann versuchen, ob
[LN1]   212:            sub   bh,2Øh
```

Listing 2. Das Assembler-Programm »mausask.asm« belebt die Maus in Batch-Dateien

```
[SA4]  213:             ;Taste Kleinbuchstabe war.
[RR1]  214:             cmp   ah,bh
[K71]  215:             je    YES
[PLØ]  216:             clc
[6V4]  217:             ;Carry löschen oder setzen...
[3D3]  218:             jmp   short CMP_END
[R55]  219:             ;...je nachdem, ob Taste gültig
[DS1]  22Ø: YES:        stc
[KA3]  221:             ;oder nicht gültig
[Z42]  222: CMP_END:    ret
[DA1]  223:             ;Fertig
[ØØØ]  224:
[ØØØ]  225:
[DU1]  226: FLAG        db  Ø
[RM3]  227:             ;Maus_vorhanden_Flag
[DZ1]  228: PARAM       dw  Ø
[CN3]  229:             ;Anfang der Parameter
```

```
[P82]  23Ø:             ;(ohne "Text",)
[LM4]  231: USAGE_TXT db  "MAUSASK v1.ØØ, "
[RS4]  232:           db  "(c) 1991 DOS International"
[9N1]  233:           db  Ødh,Øah
[Y25]  234:           db  "    Autor: Jochen Bleichroth
       "
[9N1]  235:           db  Ødh,Øah
[9N1]  236:           db  Ødh,Øah
[AD3]  237:           db  "Aufruf: mausask "
[DZ2]  238:           db  '["Text",]Bild'
[XG3]  239:           db  "schirmzeile(a..y)"
[4X1]  24Ø:           db  "Taste..."
[7A3]  241: CR_TXT    db  Ødh,Øah,"$"
[ØØØ]  242:
[BW1]  243: CODE      ends
[4W1]  244:           end     START
```

Listing 2. Das Assembler-Programm »mausask.asm« belebt die Maus in Batch-Dateien

Turbo C/C++ und Zortech C/C++

Fremdsprachen

Ob Englisch, Deutsch oder Französisch – ein Programm für alle Computeranwender nutzbar zu machen, dafür reicht eine Sprache meist nicht aus. Zumindest die wichtigsten Texte, etwa die Menüs und Abfragen, sollten Sie an die jeweilige Landessprache anpassen.

Die Grundidee besteht darin, die Texte vom Programm zu trennen, und in einer oder mehreren, separaten Dateien zu speichern. Die Vorteile sind eindeutig. Da die Texte in speziellen Textdateien gesammelt werden, muß der Programmierer das Programm bei einer Sprachanpassung nicht C-Modul für C-Modul nach Texten durchsuchen. Außerdem läßt sich somit die Sprache des Programms verändern, ohne daß es neu kompiliert werden muß. Es kann sogar jeder Anwender das Programm an seine persönlichen, sprachlichen Wünsche und Bedürfnisse adaptieren.
Eine eigene Funktionsbibliothek übernimmt das Laden dieser Dateien und die Zugriffe auf die gewünschten Texte mittels ID-Nummer. Ein ähnliches Verfahren wenden übrigens auch MS-Windows und nahezu alle anderen GUI (Graphical User Interface) an, um beispielsweise Texte und Bitmaps nachzuladen.
Bei dem hier vorgestellten Verfahren sind drei Schritte nötig, damit Ihr Programm eine beliebige Sprache »verstehen« kann.
Zunächst legen Sie alle Texte in einer Datei mit der Endung »ltx« (Language Text) ab. Vor jedem Text beziehungsweise jeder Textzeile setzen Sie ein C-kompatibles Label, gefolgt von einem Doppelpunkt. Dazu ein Beispiel:

```
HELLO:Welcome to the Pleasure Dome!
QUIT:Do you really wanna quit?
```

Schritt zwei besteht darin, aufgrund dieser Datei eine L- und eine H-Datei zu erzeugen. In der Datei mit der Endung »l« liegen die Texte mit einem Dateikopf zum schnellen Zugriff. Die H-Datei enthält Define-Zeilen, die den von Ihnen angegebenen Labels einen Wert zuordnen. Diese Header-Datei müssen Sie in Ihrem C-Programm einbinden.
Zum Generieren der beiden Dateien dient das Hilfsprogramm »makelang.c« (Listing 1)
Sie rufen es von der Kommandozeile auf und übergeben Name und Endung der Ltx-Datei. Darauf folgt der Name einschließlich Endung der L-Datei und dann der Name der LH-Datei samt Endung.

Ein Beispiel:

```
makelang test.ltx test.l test.lh
```

Der dritte und letzte Schritt zum Einbinden in Ihre eigenen Programme besteht darin, daß Sie das C-Modul »lang.c« (Listing 3) kompilieren und stets an Ihre Programme anhängen. Darin finden Sie alle nötigen Funktionen zum Umgang mit den Textdateien. Die Header-Datei »lang.h« (Listing 2) ist ebenfalls wichtig. Sie enthält die Prototypen der in »lang.c« definierten Funktionen. »langtest.c« (Listing 4) schließlich ist ein kleines Beispiel, wie Ihr Programm die Texte ansprechen kann.
Die Funktionen verwalten bis zu 25 verschiedene Dateien. Beim Laden einer Textdatei erhalten Sie deshalb eine ID-Nummer zurück. Diese übergeben Sie bei jedem Zugriff auf die Texte, damit die Funktionen wissen, auf welche der geladenen Textdateien sie sich beziehen.
Im folgenden die Funktionen des einzelnen Moduls:

```
void InitLanguageLib(void)
```

Rufen Sie diese Funktion zu Beginn Ihres Programms auf, damit alle Variablen ordnungsgemäß initialisiert sind.

```
int LoadLanguageFile(charptr name)
```

Mit dieser Funktion laden Sie eine Textdatei mit dem Namen »name« ein. »name« muß den Namen sowie die Endung enthalten. Als Ergebnis erhalten Sie die ID-Nummer der Datei oder eine Null für einen Fehler zurück.

```
void FreeLanguageFile(int ID)
```

Benötigen Sie eine Textdatei nicht mehr, so sollten Sie diese freigeben. Dazu rufen Sie diese Funktion auf und übergeben als Parameter die ID-Nummer der Datei.

```
void FreeAllLanguageFiles(void)
```

Benötigen Sie keine der geladenen Textdateien mehr, so entfernen Sie mit dieser Funktion alle Texte aus dem Speicher.

```
int LGetTxtLen(int fileID,int txtID)
```

Diese Funktion gibt die Länge des Textes mit der ID »txtID« in der geladenen Textdatei mit der ID »fileID« zurück.

```
int LGetTxt(int fileID,int txtID,charptr str)
```

Der durch »txtID« und »fileID« spezifizierte Text wird in den Puffer »str« kopiert. Bei Erfolg ist der Rückgabewert 1, ansonsten Null. Beachten Sie, daß der Text in den Puffer passen muß. Sie können etwa den Puffer zuvor dynamisch erzeugen, indem Sie mittels »LGetTxtLen« die Textlänge abfragen.

```
charptr LGetTxtAdr(int fileID,int txtID)
```

Diese Funktion liefert lediglich die Adresse des gesuchten Textes zurück. Dieser ist stets Null-terminiert.

Zum Schluß noch einige Anmerkungen

Sollten Sie statt dem hier verwendeten Speichermodell »Small« ein anderes Modell verwenden, so müssen Sie gegebenenfalls in der Header-Datei die Typendeklaration von »charptr« sowie den Kern von »LGetTxtAdr()« ändern.

(Martin Backschat/hs)

```
[WP3]    1: /* Programm: makelang.c
[826]    2:    Funktion: generiert L- und Lh-Dateien
[GR5]    3:    Sprache:  Turbo C++ und Zortech C++
[774]    4:    Autor:    Martin Backschat
[7H4]    5:    (c) DMV Widuch GmbH & Co. KG */
[ØØØ]    6:
[XD6]    7: #define DLEN 1ØØØØ /* Max. Tabellgroesse */
[TA6]    8: #define TLEN 3ØØØØ /* Max. Textgroesse   */
[7W6]    9: #include <stdio.h> /* Standard-Includes  */
[U83]   1Ø: #include <stdlib.h>
[U55]   11: typedef struct { /* Dateikopf */
[5L4]   12:   unsigned id,filelen,count;
[VY1]   13: } Langhead;
[ØØØ]   14:
[UX4]   15: void main(int argc,char **argv)
[P7Ø]   16: {
[2S3]   17:   FILE *src,*dest1,*dest2;
[792]   18:   Langhead head;
[TØ2]   19:   char *tptr,c;
[ZL5]   2Ø:   unsigned int *dptr,textnr,offset,
[423]   21:                 oldoffset,cnt,len;
[ØØØ]   22:
[5C6]   23:     printf("\nLanguage Module Generator \
[CC2]   24: v2 (2Ø.12.9Ø)\n");
[DR3]   25:     if (argc != 3 && argc != 4) {
[VM5]   26:         printf("\nParameter: ASCII-File \
[NC5]   27: Module-File [Include-File]\n");
[961]   28:         exit(Ø);
[U7Ø]   29:     }
[KE4]   3Ø:     /* .ltxt-Textdatei oeffnen */
[4C5]   31:     if (!(src = fopen(argv[1],"rt"))) {
[6R4]   32:         printf("\nCannot open %s!\n",
[6K1]   33:                 argv[1]);
[961]   34:         exit(Ø);
[U7Ø]   35:     }
[Ø95]   36:     /* .l-Binaerdatei neu erstellen */
[1M5]   37:     if (!(dest1 = fopen(argv[2],"wb"))) {
[635]   38:         printf("\nCannot create %s!\n",
[9K1]   39:                 argv[2]);
[961]   4Ø:         exit(Ø);
[U7Ø]   41:     }
[BR5]   42:     /* Falls angeg., .lh-Datei erzeugen */
[HY1]   43:     if (argc == 4)
[ZS1]   44:         if (!(dest2 =
[DT3]   45:             fopen(argv[3],"wt"))) {
[635]   46:             printf("\nCannot create %s!\n",
[CK1]   47:                     argv[3]);
[961]   48:             exit(Ø);
[U7Ø]   49:         }
[SG3]   5Ø:     if (!(dptr = (unsigned *)
[8V3]   51:         malloc((size_t)DLEN))) {
[EVØ]   52: err2:
[2K5]   53:         printf("\nNot enough memory!\n");
[961]   54:         exit(Ø);
[U7Ø]   55:     }
[1U2]   56:     if (!(tptr = (char *)
[XL3]   57:         malloc((size_t)TLEN)))
[SP1]   58:         goto err2;
[ØØØ]   59:
[T65]   6Ø:     textnr = offset = oldoffset = Ø;
[HY1]   61:     if (argc == 4)
[ZW4]   62:         fprintf(dest2,"/*\tLanguage \
[XV4]   63:             Module <%s>\t*/\n\n",argv[2]);
[ØØØ]   64:
[FU1]   65:     while (1) {
[DF5]   66:         /* Labelbezeichnung einlesen ..: */
[PY4]   67:         while (!feof(src)) { /* bei EOF */
[4R3]   68:             if ((c = tptr[offset] =
[FM2]   69:                 fgetc(src)) == ':')
[PØ1]   7Ø:                 break;
[ED1]   71:             offset++;
[U7Ø]   72:         }
[9E1]   73:         if (c != ':')
[Y14]   74:             break;  /* -> eof erreicht */
[392]   75:         if ((argc == 4) &&
[4P3]   76:             (oldoffset != offset)) {
[UV2]   77:             tptr[offset] = Ø;
[2Ø4]   78:             fprintf(dest2,"#define\
[AT6]   79: \t%s\t%u\n",&tptr[oldoffset],textnr+1);
[U7Ø]   8Ø:         }
[VT4]   81:         /* Labelname ueberschreiben */
[C83]   82:         offset = oldoffset;
[ØØØ]   83:
[UØ3]   84:         while (!feof(src)) {
[4R3]   85:             if ((c = tptr[offset] =
[DF2]   86:                 fgetc(src)) == 13)
[AM1]   87:                 continue;
[8A2]   88:             else if (c == 1Ø)
[PØ1]   89:                 break;
[ED1]   9Ø:             offset++;
[U7Ø]   91:         }
[4S1]   92:         if ((c != 1Ø) &&
[ØH3]   93:             (offset == oldoffset))
[XØ3]   94:             goto eofendempty;
[H83]   95:         tptr[offset++] = Ø;
[NU4]   96:         dptr[textnr++] = oldoffset;
[T93]   97:         oldoffset = offset;
[U7Ø]   98:     }
[R62]   99: eofendempty:
[H94]  1ØØ:     len = textnr*sizeof(int)+
[PU2]  1Ø1:             sizeof(Langhead);
[Z52]  1Ø2:     head.id = 'LG';
[N54]  1Ø3:     head.filelen = offset+len;
[286]  1Ø4:     head.count = textnr; /* Anzahl texte */
[ØØØ]  1Ø5:
[ZP5]  1Ø6:     if (fwrite(&head,sizeof(Langhead),1,
[DZ1]  1Ø7:                 dest1) != 1) {
[VV1]  1Ø8: write_err:
[R45]  1Ø9:         printf("\nFile write error!\n");
[961]  11Ø:         exit(Ø);
[U7Ø]  111:     }
[U75]  112:     for (cnt = Ø; cnt < textnr; cnt++)
[KE5]  113:         dptr[cnt] += textnr*sizeof(int);
[NZ5]  114:     if (fwrite(dptr,sizeof(int),textnr,
[CL2]  115:                 dest1) != textnr)
[EX2]  116:         goto write_err;
[1C5]  117:     if (fwrite(tptr,1,offset,dest1) !=
[581]  118:         offset)
[EX2]  119:         goto write_err;
[RH5]  12Ø:     printf("\n%s gewandelt:",argv[1]);
[TH6]  121:     printf("\tText: %u\tStrings: %u\n\n",
[ØN2]  122:             offset,textnr);
[C61]  123:     exit(1);
[U7Ø]  124: }
```

Listing 1. »makelang.c« erzeugt aus einer Ltx-Datei die beiden L- und LH-Dateien.

```
[XM2]    1: /* Datei:    lang.h
[YS5]    2:    Funktion: Haeder-Datei für lang.c */
[ØØØ]    3:
[6K3]    4: #define charptr char *
[1SØ]    5: void
[PS3]    6:    InitLanguageLib(void),
[KN4]    7:    FreeAllLanguageFiles(void),
[G35]    8:    FreeLanguageFile(unsigned id);
[Ø52]    9: unsigned int
[H95]   1Ø:    LoadLanguageFile(charptr name),
[DD6]   11:    LGetTxt(unsigned id,unsigned textid,
[392]   12:       charptr str),
[BW6]   13:    LGetTxtLen(unsigned id,unsigned textid);
[SC1]   14: charptr
[AW6]   15:    LGetTxtAdr(unsigned id,unsigned textid);
```

Listing 2. In der Header-Datei »lang.h« sind die Funktionsprototypen des Moduls zusammengefaßt

```
[WØ3]    1: /* Programm: lang.c
[6M4]    2:    Funktion: Programmodul zur
[WK2]    3:              Texteinbindung
[GR5]    4:    Sprache:  Turbo C++ und Zortech C++
[774]    5:    Autor:    Martin Backschat
[T15]    6:    (c)1991 DMV Widuch GmbH & Co. KG */
[ØØØ]    7:
[6A3]    8: #include <string.h>
[L13]    9: #include <stdio.h>
[MK2]   1Ø: #include "lang.h"
[7G2]   11: #define MAXID 25
[3W2]   12: typedef struct {
[5L4]   13:     unsigned id,filelen,count;
[VY1]   14: } Langhead;
[XY4]   15: static unsigned *ladr[MAXID];
[ØØØ]   16:
[YE5]   17: /* Initialisiert Modul                    */
[1G4]   18: void InitLanguageLib(void)
[F91]   19: { int i;
[XU5]   2Ø:     for (i = Ø; i < MAXID; ladr[i++] = Ø);}
[ØØØ]   21:
[KØ6]   22: /* Laedt .L-Datei in den Speicher        */
[KØ5]   23: /* -> id (1..) oder Ø (Fehler)           */
[FC7]   24: unsigned int LoadLanguageFile(charptr name)
[P7Ø]   25: {
[N71]   26:   FILE *h;
[792]   27:   Langhead head;
[P91]   28:   int id;
[KT5]   29:     /* Einen freien Eintrag suchen       *
        /
[6D4]   3Ø:     for (id = Ø; id < MAXID; id++)
[C62]   31:         if (!ladr[id])
[PØ1]   32:             break;
[XW2]   33:     if (id == MAXID ||
[Z93]   34:         (!(h = fopen(name,"rb"))))
[CE1]   35:         return Ø;
[E36]   36:     /* Pruefen, ob Dateikennung stimmt   */
[EU5]   37:     if ((fread(&head,sizeof(Langhead),1,h)
[CW3]   38:         != 1) || (head.id != 'LG')) {
[PL2]   39: err:    fclose(h);
[CE1]   4Ø:         return Ø;
[U7Ø]   41:     }
[6A5]   42:     head.filelen -= sizeof(Langhead);
[XA4]   43:     if (!(ladr[id] = (unsigned *)
[KG5]   44:         malloc(head.filelen+sizeof(int))))
[TH1]   45:         goto err;
[W54]   46:     ladr[id][Ø] = head.count;
[ES3]   47:     if (fread(&ladr[id][1],
[SF3]   48:         head.filelen,1,h) != 1) {
[EW3]   49:         FreeLanguageFile(id+1);
[TH1]   5Ø:         goto err;
[U7Ø]   51:     }
[MH1]   52:     fclose(h);
[RV1]   53:     return id+1;
[U7Ø]   54: }
[ØØØ]   55:
[MB6]   56: /* Alle offenen .L-Dateien freigeben     */
```

```
[MD5]   57: void FreeAllLanguageFiles(void)
[P7Ø]   58: {
[F11]   59:   int i;
[WW3]   6Ø:     for (i = 1; i <= MAXID; i++)
[9B3]   61:         FreeLanguageFile(i);
[U7Ø]   62: }
[ØØØ]   63:
[8M5]   64: /* .L-Datei mit id freigeben              */
[NN5]   65: void FreeLanguageFile(unsigned id)
[P7Ø]   66: {
[1Z1]   67:     if (--id >= Ø &&
[AU3]   68:         id < MAXID && ladr[id]) {
[NM2]   69:         free(ladr[id]);
[SY1]   7Ø:         ladr[id] = Ø;
[U7Ø]   71:     }
[U7Ø]   72: }
[ØØØ]   73:
[W37]   74: /* Gibt Adresse d. Strings textid zurück */
[AF5]   75: charptr LGetTxtAdr(unsigned id,
[XU2]   76:                     unsigned textid)
[P7Ø]   77: {
[ST4]   78:     if (ladr[id-1][Ø] <= textid-1)
[CE1]   79:         return Ø;
[KE5]   8Ø:     return ((charptr)&ladr[id-1][1])+
[UL3]   81:             ladr[id-1][textid];
[U7Ø]   82: }
[ØØØ]   83:
[9W6]   84: /* Kopiert .L-String in angegeb. String  */
[K56]   85: unsigned int LGetTxt(id,textid,str)
[5E3]   86:   unsigned id,textid;
[X62]   87:   charptr str;
[P7Ø]   88: {
[R62]   89:   charptr ptr;
[ØØØ]   9Ø:
[NG5]   91:     if (!(ptr = LGetTxtAdr(id,textid)))
[CE1]   92:         return Ø;
[6R2]   93:     strcpy(str,ptr);
[DE1]   94:     return 1;
[U7Ø]   95: }
[ØØØ]   96:
[EH6]   97: /* Ermittelt Laenge d. Strings textid    */
[MY5]   98: unsigned int LGetTxtLen(id,textid)
```

```
[5E3]   99:   unsigned id,textid;
[P7Ø]  1ØØ: {
[R62]  1Ø1:   charptr ptr;
[NG5]  1Ø2:     if (!(ptr = LGetTxtAdr(id,textid)))
[CE1]  1Ø3:         return Ø;
[Z93]  1Ø4:     return strlen(ptr);
[U7Ø]  1Ø5: }
```

Listing 3. Kompilieren Sie das C-Modul »lang.c«, und hängen Sie es beim Linken an Ihr Programm an

```
[7T3]    1: /* Programm: langtest.c
[7M6]    2:    Funktion: demonstriert besondere Text-
[HT3]    3:              zuführung aus lang.c */
[ØØØ]    4:
[T12]    5: /* test.ltx :
[MU5]    6: HELLO:Welcome to the Pleasure Dome
[Z85]    7: QUIT:Do you really wanna quit?
[17Ø]    8: */
[ZU2]    9: #include "test.lh"
[MK2]   1Ø: #include "lang.h"
[L13]   11: #include <stdio.h>
[ØØØ]   12:
[6M1]   13: int main()
[P7Ø]   14: {
[CL2]   15:   char text[1ØØ];
[GG3]   16:   unsigned int id1,len;
[942]   17:   charptr adr;
[ØØØ]   18:
[223]   19:     InitLanguageLib();
[4F1]   2Ø:     if (!(id1 =
[EE4]   21:         LoadLanguageFile("test.l")))
[8R3]   22:         return; /* Ladefehler! */
[326]   23:     printf("%s\n",LGetTxtAdr(id1,HELLO));
[L14]   24:     LGetTxt(id1,QUIT,text);
[D93]   25:     printf("%s\n",text);
[FM4]   26:     len = LGetTxtLen(id1,QUIT);
[4V3]   27:     printf("len:%u\n",len);
[ZY3]   28:     FreeAllLanguageFiles();
[U7Ø]   29: }
```

Listing 4. Das kleine Demo-Programm »langtest.c« zeigt die Vorteile des Konzepts

Globale DOS-Operation

Es ist manchmal angebracht, vor dem Abschalten des Rechners alle Dateien auf »read only« zu setzen. Darüber hinaus gibt es noch andere DOS-Operationen, die man nur mit großem Aufwand in allen Verzeichnissen gleichzeitig ausführen kann. »all« vor den DOS-Befehl gesetzt, sorgt für die Bearbeitung des gesamten aktuellen Laufwerks.

Das Programm »all.pas« (Listing) ermöglicht die Ausdehnung von MS-DOS-Befehlen (del, copy, etc.) auf ein komplettes Laufwerk mit allen Unterverzeichnissen. Am erwähnten Beispiel »read only« erkennt man den enormen Vorteil dieses Hilfsprogramms. Setzt man das Attribut einer Datei auf » + r«, ist keine Veränderung derselben mehr möglich. Um alle Dateien auf einer Festplatte auf »read only« zu setzen, lautet die Syntax:

```
all attrib +r *.*
```

Ohne »all« müßte man in alle Verzeichnisse gehen und dort jeweils obigen Befehl (natürlich ohne »all«) eingeben, oder alle Pfadnamen im Kopf haben und sie als Parameter angeben – sicher keine leichte Aufgabe! Die ebenso einfache Umkehr des Vorgang wird erforderlich, wenn man wieder Dateien bearbeiten will:

```
all attrib -r *.*
```

»all« läßt sich auf alle MS-DOS-Befehle anwenden, deren globale Funktion Sinn macht. So funktioniert »all copy *.bak d:\ablage« ebenso wie der Befehl »all del *.bak«. Den letzteren sollte man sich jedoch vor der Eingabe gut überlegen. Sicher werden auch Sie eine Anwendung finden, die Ihnen den alltäglichen Umgang mit Ihren Dateien durch »all« erleichtert. (Alexander Platte/hs)

```
[ØF3]    1: { Programm: all.pas
[3Ø6]    2:   Funktion: trägt Befehle durch alle
[PB2]    3:             Verzeichnisse
[YY5]    4:   Sprache:  Turbo Pascal ab Version 5.Ø
[S74]    5:   Autor:    Alexander Platte
[P35]    6:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    7:
[912]    8: {$M 5ØØØØ, Ø, Ø}
[ØØØ]    9:
[S62]   1Ø: USES Dos,Crt;
[ØØØ]   11:
[RNØ]   12: VAR
[WZ2]   13:   befehl    : STRING;
[TX3]   14:   ursprung  : STRING[8];
[ØØØ]   15:
[ZK3]   16: PROCEDURE fuehreaus;
[MUØ]   17: BEGIN
[N72]   18:   SwapVectors;
[MØ6]   19:   Exec(GetEnv('Comspec'), '/C ' + befehl);
[N72]   2Ø:   SwapVectors;
[ZNØ]   21: END;
[ØØØ]   22:
[YN5]   23: PROCEDURE suchedir(pfad : STRING);
[RNØ]   24: VAR
[7Y2]   25:   datei : searchrec;
[MUØ]   26: BEGIN
[ZØ5]   27:   FindFirst(pfad + '*.*', $12, datei);
[H63]   28:   WHILE doserror = Ø DO
[MUØ]   29:     BEGIN
[6D4]   3Ø:       IF (datei.attr AND $1Ø = $1Ø) AND
```

```
[6Y3]   31:          (datei.name[1] <> '.') THEN
[MUØ]   32:       BEGIN
[XT3]   33:         ChDir(pfad + datei.name);
[NX4]   34:         WriteLn(#13, pfad + datei.name);
[AS1]   35:         fuehreaus;
[YX4]   36:         suchedir(pfad+datei.name+'\');
[ZNØ]   37:       END;
[UN2]   38:     FindNext(datei);
[ZNØ]   39:   END;
[ZNØ]   4Ø: END;
[ØØØ]   41:
[MUØ]   42: BEGIN
[YX4]   43:   befehl := Concat(ParamStr(1), ' ',
[DX4]   44:   ParamStr(2), ' ', ParamStr(3), ' ',
[G22]   45:   ParamStr(4));
[KZ2]   46:   IF befehl = '   ' THEN
[MUØ]   47:     BEGIN
[R16]   48:       WriteLn('Syntax: ALL [DOS-Befehl]',
[AEØ]   49:               #7);
[T41]   5Ø:       Halt(1);
[ZNØ]   51:     END;
[A93]   52:   GetDir(Ø, ursprung);
[ØM1]   53:   ChDir('\');
[AS2]   54:   WriteLn(#13, '\');
[AS1]   55:   fuehreaus;
[492]   56:   suchedir('\');
[WR2]   57:   ChDir(ursprung);
[KNØ]   58: END.
```

Listing. »all.pas« führt DOS-Befehle in allen Verzeichnissen aus

Turbo Pascal ab Version 4.0

Matt dem Schnittstellenchaos

Leider liegt dem BIOS nur die Information über zwei serielle Schnittstellen vor. Wird eine weitere Schnittstellenkarte eingebaut, kann das Betriebssystem ohne Aktivierung von »com3« und »com4« diese Ports nicht nutzen. Die Lösung des Problems heißt »addcom«.

Oft steht man vor der Situation, daß man mit zwei seriellen Schnittstellen nicht mehr auskommt. Eine wird für die Maus benötigt, eine für den Drucker, eine weitere für das Modem etc. Was läge da näher, als einfach eine Zusatzkarte mit seriellen Schnittstellen in den Computer einzubauen, wäre da nicht das BIOS, das lediglich zwei serielle Schnittstellen unterstützt.

Glücklicherweise wurde im BIOS-Datensegment schon Platz für vier serielle Ports reserviert, so daß hier nur noch bis zu zwei weitere Schnittstellen »angemeldet« werden müssen.

Das Turbo-Pascal-Programm »addcom« (Listing) übernimmt hierbei sozusagen die Anmeldeformalitäten.
Der Aufruf des Programms erfolgt mit

```
addcom COMn=mmmm
```

wobei »n« für die Nummer der seriellen Schnittstelle steht und »mmmm« für die Portadresse, an der das Betriebssystem die Schnittstelle finden kann. Die Basisadresse kann dabei in dezimaler oder hexadezimaler Schreibweise übergeben werden, beispielsweise

```
addcom COM3=1000
```

in dezimaler Schreibweise oder

```
addcom COM3=$3E8 COM4$2E8
```

in hexadezimaler Notation.
Während der Installation prüft das Programm, ob überhaupt an der angegebenen Stelle eine passende Karte vorhanden ist. Sofern die Karte nicht gefunden wird, setzt das Programm den MS-DOS-Fehlercode auf 1, so daß in einer aufrufenden Stapeldatei mit »errorlevel« darauf reagiert werden kann.
Nach erfolgter Zuweisung von »com3« und »com4« läßt sich das Betriebssystem dazu bewegen, mit »mode« die neue(n) Schnittstelle(n) zu konfigurieren, beispielsweise durch

```
mode com4:9600,n,8,1
```

Auf einen Test, ob die Schnittstellen auch an den richtigen Basisadressen liegen, wurde verzichtet, damit sich die Adressen auch problemlos tauschen, beziehungsweise sich auch Karten an »Nicht-Standard-Adressen« betreiben lassen.

(Marcus P. Gau/bj)

```
[PS3]    1: { Programm: addcom.pas
[UP5]    2:   Funktion: trägt Com3 und Com4 ein
[UY5]    3:   Sprache:  Turbo Pascal ab Version 4.Ø
[8R3]    4:   Autor:    Marcus P. Gau
[P35]    5:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    6:
[P34]    7:   Syntax : ADDCOM COMn=mmm
[DV4]    8:    n    = Nummer der Schnittstelle
[CY5]    9:    mmm = dezimale Adresse der Karte oder
[376]   1Ø:    $mmm = hexadezimale Adresse derselben
[U7Ø]   11: }
[ØØØ]   12:
[6R2]   13: PROGRAM Addcom;
[RNØ]   14: VAR
[W55]   15:   ComAddrA  : ARRAY [1..4] OF WORD
[5Ø3]   16:               ABSOLUTE $ØØ4Ø:$ØØØØ;
[DP2]   17:   Hardware2 : BYTE
[AØ3]   18:               ABSOLUTE $ØØ4Ø:$ØØ11;
[WB2]   19:   Coms      : BYTE;
[TV2]   2Ø:   ComNumber : BYTE;
[9R2]   21:   ComAddr   : WORD;
[6C3]   22:   Parameter : STRING;
[873]   23:   ComToAdd  : STRING;
[K73]   24:   AddrStr   : STRING;
[MØ2]   25:   I         : BYTE;
[T83]   26:   ValCode   : INTEGER;
[SF3]   27:   InstError : BOOLEAN;
[ØØØ]   28:
[W25]   29: FUNCTION Upper(InStr : STRING)
[8F1]   3Ø:           : STRING;
[RNØ]   31: VAR
[SW2]   32:   Len       : INTEGER;
[BL2]   33:   I         : INTEGER;
[6A2]   34:   InChr     : CHAR;
[193]   35:   HelpStr   : STRING;
[MUØ]   36: BEGIN
[AW2]   37:   HelpStr := InStr;
[U34]   38:   Len     := Length(HelpStr);
[DV2]   39:   FOR I := 1 TO Len DO
[MUØ]   4Ø:   BEGIN
[1R3]   41:     InChr       := HelpStr[I];
[H14]   42:     InChr       := UpCase(InChr);
[MD3]   43:     HelpStr[I] := InChr;
[ZNØ]   44:   END;
[123]   45:   Upper   := HelpStr;
[ZNØ]   46: END;
[ØØØ]   47:
[A66]   48: FUNCTION IsPortPresent(Addr : WORD)
```

```
[FK1]   49:           : BOOLEAN;
[RNØ]   5Ø: VAR
[FN2]   51:   DummyReg : WORD;
[MUØ]   52: BEGIN
[B53]   53:   DummyReg := Addr + 7;
[8M3]   54:   Port[DummyReg] := $ØØ;
[6K4]   55:   IF Port[DummyReg] = $ØØ THEN
[MUØ]   56:   BEGIN
[AP3]   57:     Port[DummyReg] := $FF;
[YU2]   58:     IsPortPresent :=
[TX3]   59:       (Port[DummyReg] = $FF);
[4MØ]   6Ø:   END
[ZRØ]   61:   ELSE
[114]   62:     IsPortPresent := False;
[ZNØ]   63: END;
[ØØØ]   64:
[MUØ]   65: BEGIN
[Z83]   66:   InstError := False;
[DM3]   67:   IF ParamCount >= 1 THEN
[MUØ]   68:   BEGIN
[9Ø5]   69:     Coms := (Hardware2 AND $Ø6) shr 1;
[DB4]   7Ø:     FOR I := 1 TO ParamCount DO
[MUØ]   71:     BEGIN
[144]   72:       Parameter := ParamStr(I);
[433]   73:       ComToAdd  := Upper(
[F33]   74:                      Copy(Parameter,1,5)
[38Ø]   75:                           );
[Z65]   76:       IF Copy(ComToAdd,1,3) = 'COM' THEN
[MUØ]   77:       BEGIN
[7PØ]   78:         Val(
[WUØ]   79:           Copy(
[5W1]   8Ø:             ComToAdd,4,1
[P6Ø]   81:               ),
[ZZ2]   82:           ComNumber,ValCode
[38Ø]   83:            );
[BH3]   84:         IF (ComNumber >= 1) AND
[Z73]   85:            (ComNumber <= 4) THEN
[MUØ]   86:         BEGIN
[M72]   87:           AddrStr := '$' +
[C53]   88:                      Copy(Parameter,6,4);
[W15]   89:           Val(AddrStr,ComAddr,ValCode);
[AU1]   9Ø:           Write('com',
[VR1]   91:                 ComNumber,
[DEØ]   92:                 '=',
[6E1]   93:                 AddrStr,
[732]   94:                 ' (',ComAddr,')'
[38Ø]   95:                );
```

Listing. »addcom.pas« aktiviert die seriellen Schnittstellen Com3 und Com4 zur weiteren Verwendung

```
[745]   96:            IF IsPortPresent(ComAddr) THEN
[MUØ]   97:            BEGIN
[XØ4]   98:              ComAddrA[ComNumber] :=
[ZD1]   99:                ComAddr;
[DN1]  1ØØ:              Inc(Coms);
[XF1]  1Ø1:              WriteLN;
[4MØ]  1Ø2:            END
[ZRØ]  1Ø3:            ELSE
[MUØ]  1Ø4:            BEGIN
[M24]  1Ø5:              WriteLN(' not present !');
[G33]  1Ø6:              InstError := True;
[4MØ]  1Ø7:            END
[4MØ]  1Ø8:          END
[ZRØ]  1Ø9:          ELSE
[MUØ]  11Ø:          BEGIN
[AF1]  111:            WriteLN(
[YB4]  112:              'No aditional COM-number ',
[VR1]  113:              ComNumber,
[TAØ]  114:              ' !'
[38Ø]  115:                  );
[G33]  116:            InstError := True;
[ZNØ]  117:          END;
[4MØ]  118:        END
[ZRØ]  119:        ELSE
[MUØ]  12Ø:        BEGIN
[FE5]  121:          WriteLN('Error in Parameter ',I);
[ZNØ]  122:        END;
```

```
[ZNØ]  123:      END;
[DG1]  124:      Coms := Ø;
[2G2]  125:      FOR I := 1 TO 4 DO
[MUØ]  126:      BEGIN
[444]  127:        IF ComAddrA[I] <> $ØØØØ THEN
[DN1]  128:          Inc(Coms);
[ZNØ]  129:      END;
[4G2]  13Ø:      WriteLN ('Now ',
[KUØ]  131:               Coms,
[SW3]  132:               ' COM-ports installed');
[624]  133:      IF Coms = 4 THEN Coms := 3;
[Ø34]  134:      Hardware2 := Hardware2 AND
[THØ]  135:                   $F9;
[AX3]  136:      Hardware2 := Hardware2 OR
[RW1]  137:                   (Coms shl 1);
[4MØ]  138:    END
[ZRØ]  139:    ELSE
[MUØ]  14Ø:    BEGIN
[U25]  141:      WriteLN('no COM-ports added');
[G33]  142:      InstError := True;
[ZNØ]  143:    END;
[WX2]  144:    IF InstError THEN
[T41]  145:      Halt(1);
[KNØ]  146: END.
[ØØØ]  147:
```

Listing. »addcom.pas« aktiviert die seriellen Schnittstellen Com3 und Com4 zur weiteren Verwendung

Wochentags-Berechnung – einmal völlig anders

Wenn man für ein bestimmtes Datum in einem Programm den entsprechenden Wochentag ermitteln will, werden dafür meist sehr aufwendige und umfangreiche Programmroutinen angewendet. Dabei ist es mit einem überraschend einfachen Trick möglich, dies »fast« ohne Programmmieraufwand zu bewerkstelligen.

Bei der Unit »wtag.pas« (Listing 1) handelt es sich um eine extrem einfache Wochentagsberechnung. Der dabei angewandte Trick ist, sich einfach der Systemuhr zu bedienen! Durch das Setzen, Zwischenspeichern und wieder Zurücksetzen des Systemdatums »mißbraucht« man die interne Uhr für eigene Zwecke und erspart sich dadurch umständliche Berechnungen. Daß dafür nur der Zugriff auf einen begrenzten Datumsbereich (01.01.1980 bis zum 31.12.2199) möglich ist, dürfte dabei in der Regel zu verschmerzen sein. Wer schon eine Wochentagsberechnung für alle Jahre seit Papst Gregor dem »ich weiß nicht wievielten« programmiert oder sonstwo abgeschrieben hat, wird sicher sagen »brauche ich nicht«.

An diesem Problem scheitern auch schon die meisten Berechnungen, allenfalls unter Unix bekommen Sie über das Cal-Programm korrekte Ergebnise. Das erfordert wiederum sehr aufwendige Programmierung und es stellt sich die Frage: »Wie sinnvoll ist die Berechnung eines Wochentags von 1900?«

Hier sollte aber nicht nur der Verwendungszweck im Vordergrund stehen, sondern vorrangig die Idee einer Problemlösung als solche.

Diese ist dabei ebenso einfach wie genial. Der Funktion »getwt« wird das Datum in Form dreier Variablen vom Typ »Word« übergeben, wie sie auch von der Funktion »GetDate« der Turbo-Pascal-Unit »Dos« zurückgeliefert werden. Das Funktionsergebnis ist eine Variable vom Typ »Byte«, welche die Wochentagsnummer enthält (0 = Sonntag, 1 = Montag, ... , 6 = Samstag). Über diese läßt sich nun mit »case« der entsprechende Wochentag in der gewünschen Schreibweise zuweisen.

Um die Arbeitsweise der Unit zu veranschaulichen, können Sie sich mit dem Demo-Program »demo.pas« (Listing 2) noch einmal den historischen Tag des Mauerfalls zwischen den beiden Teilen Deutschlands anzeigen lassen. Sollten sie hier nicht »Donnerstag« erhalten, kann das zwei Ursachen haben. Entweder ist in Ihnen beim Abtippen des Programms ein Fehler unterlaufen, oder, was unwahrscheinlich ist, Ihre Systemuhr ist defekt.

(Ingo Rau/hs)

```
[5G3]    1: { Programm: wtag.pas
[DL5]    2:   Funktion: Wochentagsberechnung }
[ØØØ]    3:
[1T1]    4: UNIT wtag;
[ØØØ]    5:
[SK1]    6: INTERFACE
[ØØØ]    7:
[VG1]    8: USES Dos;
[ØØØ]    9:
[5Z5]   1Ø: FUNCTION getwt (j,m,t : WORD) : BYTE;
[DE6]   11: { Ermittelt die Wochentagsnummer  }
[ØØØ]   12:
[4L2]   13: IMPLEMENTATION
[ØØØ]   14:
[6T2]   15: FUNCTION getwt;
[HG4]   16:   VAR aj,am,at,awt,wt : WORD;
[ØØØ]   17:
[MUØ]   18:   BEGIN
[UR3]   19:     GetDate (aj,am,at,awt);
[TG2]   2Ø:     SetDate (j,m,t);
[S13]   21:     GetDate (j,m,t,wt);
[B43]   22:     SetDate (aj,am,at);
[1Ø2]   23:     getwt := wt;
[ZNØ]   24:   END;
[ØØØ]   25:
[KNØ]   26: END.
```

Listing 1. Die Unit »wtag.pas« dient zur Berechnug des Wochentages für ein bestimmtes Datum

```
[AF3]    1: { Programm: demo.pas
[7X4]    2:   Funktion: Datumsberechnung }
[ØØØ]    3:
[VG1]    4: USES Dos;
[ØØØ]    5:
[8P2]    6: VAR aj,at,am,awt,
[TY2]    7:     j,m,t,wt       : WORD;
[6B3]    8:     wtstr          : STRING;
[ØØØ]    9:
[MUØ]   1Ø:   BEGIN
[1S3]   11:     GetDate (aj,at,am,awt);
[7W2]   12:     SetDate (1989,11,9);
[S13]   13:     GetDate (j,m,t,wt);
[K43]   14:     SetDate (aj,at,am);
[ZR1]   15:     CASE wt OF
[HM3]   16:       Ø : wtstr := 'Sonntag';
[RA3]   17:       1 : wtstr := 'Montag';
[FM3]   18:       2 : wtstr := 'Dienstag';
[RR3]   19:       3 : wtstr := 'Mittwoch';
[F14]   2Ø:       4 : wtstr := 'Donnerstag';
[9K3]   21:       5 : wtstr := 'Freitag';
[WM3]   22:       6 : wtstr := 'Samstag';
[ZNØ]   23:     END;
[EC6]   24:   WriteLn ('Der 9.11.199Ø war ein ',wtstr);
[ØØØ]   25:
[KNØ]   26: END.
```

Listing 2. »demo.pas« zeigt die Anwendung der Unit anhand der Berechnung eines historischen Datums

DOS EXTRA



DOS EXTRA 2
3D-CAD-Programm
DOS-Commander
DOS-Utilities
SForth-Compiler

DOS EXTRA 3
Der PC – glasklar
DFÜ-Programm
Hercules-Grafik
Sprache digitalisiert

DOS EXTRA 4
Speicherresidente Programme
DTP-Programm
EGA-Grafik
MS-DOS Intern

DOS EXTRA 5
PC-Grafikkarten
Das BIOS des PC
GW-Basic
Gerätetreiber programmieren

DOS EXTRA 6
Basic im Detail
GW-Basic intern
SBasic-Compiler
Basic-Utilities

DOS EXTRA 7
PC-Komponenten
Sprite-Editor
SBasic Plus
C-Grundlagen

DOS EXTRA 8
Assembler SAM III
Assembler und Hochsprachen
Assembler-Werkstatt
Assembler-Grundlagen

DOS EXTRA 9
Alles für Einsteiger
MS-DOS auf einen Blick
Hardware im Detail
Einsteiger-Praxis

DOS EXTRA 10
MS-DOS 4.01
Hardware-Praxis
Snapshot-Programm
Turbo-Pascal-Units

DOS EXTRA 11
Turbo Pascal total
Turbo Pascal intern
SAA-Blockbefehle
MS-DOS-Speicherverwaltung

DOS EXTRA 12
Grafik in Basic
Hardcopy-Utility
Fraktalgrafik in Turbo Pascal
Basic für PC-Grafikkarten



NEU

DOS EXTRA Nr. 13

Aus dem Inhalt:

- Grundlagen: Spielekategorien
- Die besten Spiele ausführlich vorgestellt
- Tips & Tricks zu Abenteuerspielen
- Großer Sonderteil zum Flugsimulator
- Shareware- und Online-Spiele

DOS EXTRA Einzelpreis DM 18,– *

Vorsprung für PC-Benutzer mit DOS-EXTRA im Abonnement!

4 Ausgaben bei vierteljährlicher Erscheinungsweise mit dem bequemen Zustellservice per Post:

BRD	DM 60,–
Europa	DM 70,–
Außereuropäisches Ausland	DM 80,–

Keine Bestellprobleme mit dem Dauerbezug der DOS-EXTRA-DATABOX!

Bequemes Dauerbezugsverfahren mit Bankabbuchung und vierteljährlicher Kündigungsmöglichkeit:

BRD	DM 39,–
Europa je	DM 41,–
Außereuropäisches Ausland je	DM 42,–

Bitte benutzen Sie nur die Bestellkarte!

DATABOX

Die Software zu den DOS-EXTRA-Heften

Sagenhaft günstig, extra viel gute Software fürs Geld! Bringt alle Programme der DOS EXTRA auf Datenträger

Den Aufbau der Databox-Diskette finden Sie in der Datei »read.me« erklärt. Geben Sie deshalb vor dem Start »type read.me« ein. Sämtliche Pascal- und C-Programme liegen auch in einer compilierten Form für Hercules- und Farbgrafikkarten vor. Es ist allerdings nicht gewährleistet, daß diese Versionen auf Hardware, die nicht vollständig kompatibel zum IBM-PC ist, problemlos laufen. In diesem Fall müssen Sie den Quellcode nach der Anpassung mit dem entsprechenden Compiler neu übersetzen.
Die Basic-Programme arbeiten mit dem GW-Basic-Interpreter von Microsoft einwandfrei. Bei Interpretern und Compilern anderer Hersteller kann es zu Problemen bei der Programmausführung kommen, sofern diese nicht voll kompatibel sind.

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

2 Disketten 5 1/4" DM 35,– *
1 Diskette 3 1/2" DM 35,– *

* Unabhängig von der Anzahl der bestellten Produkte berechnen wir für das Inland 4,– bzw. für das Ausland 6,– DM Porto und Verpackung.

– Bitte benutzen Sie die Bestellkarte. –

DMV-Verlag • Postfach 250 • 3440 Eschwege

DMV Daten- und Medienverlag



Turbo Pascal 5.5 und MASM Version 4.0

Mehr Komfort mit SBackup

Die Daseinsberechtigung einer weiteren Backup-Lösung wird durch verschiedene, schon vorhandene in Frage gestellt. Ein bedeutender Grund ist der relativ hohe Preis professioneller Versionen. Wer auf vielfältige und wenig benötigte »Features« verzichten kann, Bedienungskomfort aber nicht missen möchte, greift zu »SBackup«.

Es sollte der Zweck eines jeden Backup-Programms sein, Daten von Festplatte auf Diskette zu übertragen, um sie bei Datenverlust schnell und unkompliziert wiederherzustellen. Hierzu wurden neben Turbo Pascal schnelle Assembler-Routinen (Listings 2 bis 6) verwendet. Mit »sbackup.pas« (Listing 1) sind Sie nun in der Lage auch nach aufgetretenen Fehlern mit dem Backup fortzufahren (Teil-Restore). Auch können einzelne Dateien oder Dateigruppen nach Belieben wiederhergestellt werden (»File Restore«). Im Zusammenhang mit der Übertragungsgeschwindigkeit von SBackup, die praktisch nur durch die Schnelligkeit der Festplatte beziehungsweise des Diskettenlaufwerks begrenzt ist, ergibt dies eine gelungene Kombination aus Datensicherheit auf der einen und Schnelligkeit auf der anderen Seite.

Die Backup-Optionen

- Uhrzeit/Datum

Ist diese Option eingeschaltet, werden nur Dateien gesichert, die nach dem angegebenen Datum erzeugt wurden. Es empfiehlt sich, zunächst ein »Grund-Backup« von der Festplatte mit ausgeschalteter Option »Uhrzeit/Datum« zu generieren und dann, in regelmäßigen Abständen, mit eingeschalteter Option »Uhrzeit/Datum« ein »Ergänzungs-Backup« aller seitdem neu hinzugekommenen beziehungsweise geänderten Dateien zu machen.

- Verify

Hat die gleiche Fubktion wie die von MS-DOS bekannte Verify-Option.

- Quellaufwerk

Vor jedem Backup müssen Sie hier das Quellaufwerk sowie die Verzeichnisse auswählen, welche die zu sichernden Daten enthalten. Dazu klickt man mit der Maus das gewünschte Verzeichnis an, oder man bringt mit Hilfe der Cursortasten den Auswahlbalken darauf und drückt anschließend [Enter] (Bild 1). Mit der Maus kann gerollt werden, indem man das Feld mit dem Pfeil anklickt, welches in die gewünschte Rollrichtung zeigt. Für jedes Verzeichnis wird an-

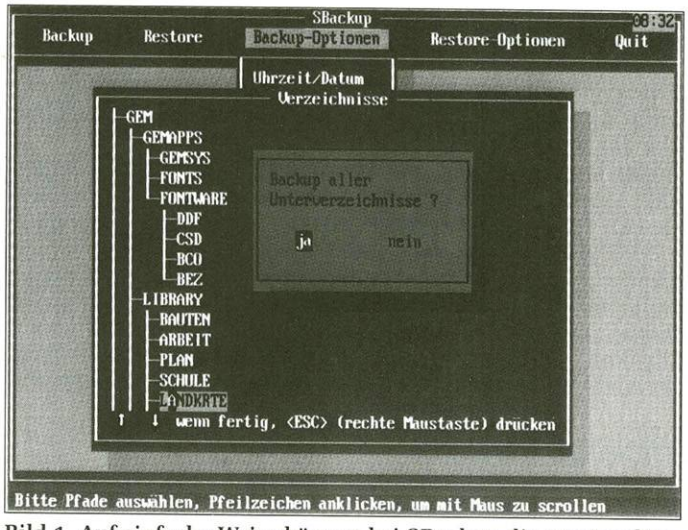


Bild 1. Auf einfache Weise können bei SBackup die gewünschten Einstellungen vorgenommen werden

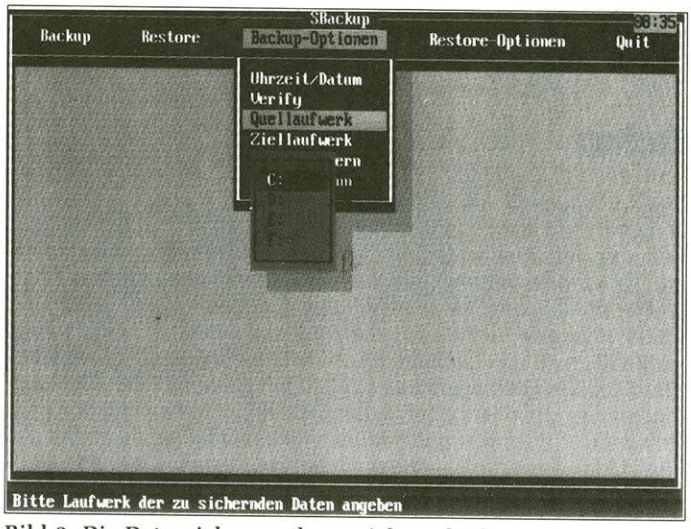


Bild 2. Die Datensicherung kann sich auch über mehrere Quelllaufwerke erstrecken

schließend abgefragt, ob alle Unterverzeichnisse mitgesichert werden sollen beziehungsweise welche Suchmaske für das Verzeichnis gelten soll (Voreinstellung ist »* . *«, die Jokerzeichen sind zu denen in MS-DOS äquivalent). Verzeichnisse, deren Unterverzeichnisse mitgesichert werden, sind mit »δ«, Verzeichnisse, deren Unterverzeichnisse nicht mitgesichert werden, mit »√« gekennzeichnet.
Es sei noch angemerkt, daß durchaus mehrere Quellaufwerke (Bild 2) und ihre Verzeichnisse ausgewählt werden können. Dazu braucht man nur den Menüpunkt »Quellaufwerk« nochmals anzuwählen und dann das gewünschte Laufwerk anzugeben.

- Ziellaufwerk

Das Laufwerk, auf dem die Daten abgespeichert werden, ist das Ziellaufwerk. Hier gibt es nur die zwei Optionen A oder B. Die Voreinstellung ist Laufwerk A.

- Konfiguration sichern

Speichert die eingestellten Quellaufwerks-Daten in einer Datei mit frei wählbarem Namen. Sie können so verschiedene Einstellungen verwalten und brauchen diese nicht jedesmal neu zu editieren.

- Konfiguration laden

Lädt eine vorher gespeicherte Quellaufwerks-Einstellung. Es werden, falls mindestens zwei vorhanden sind, alle Konfigurationsdateien angezeigt, und der Benutzer kann nun die gewünschte Datei mit Hilfe der Maus oder der Cursortasten auswählen. Konnte nur eine gefunden werden, wird diese ohne weitere Abfrage geladen.

Restore-Optionen

- Verzeichnis erzwingen

Ist diese Option eingeschaltet, werden die Verzeichnisse der gesicherten Dateien, falls nicht vorhanden, neu generiert. Bei ausgeschalteter Option »Verzeichnis erzwingen« wird der Benutzer zur Eingabe eines Verzeichnisses aufgefordert, wenn das ursprüngliche Verzeichnis nicht vorhanden ist. Voreinstellung ist hierbei das aktuelle Verzeichnis.

- Verify

Hat die gleiche Funktion wie bei Backup-Optionen.

- Quellaufwerk

Analoge Funktion wie bei Backup-Optionen, allerdings zum Lesen der gesicherten Daten.

Das Restore-Menü

- Restore

»Restore« bricht bei Auftreten eines Schreibfehlers nicht ab; die zu schreibende Datei wird allerdings nicht wiederhergestellt. Damit der Benutzer weiß, um welche Datei es sich handelt, erfolgt ein entsprechender Hinweis. Bei Auftreten eines Lesefehlers auf der Diskette bricht Restore jedoch die weitere Ausführung sofort ab. Sie können dann mit »Teilrestore« oder »Filerestore« versuchen, wenigstens einen Teil Ihrer Daten zu retten.

- Teil-Restore

Funktioniert wie Restore, die Ausführung beginnt aber nicht mit Diskette 1, sondern eine beliebige Diskette kann als erste Diskette genommen werden. »Teilrestore« gibt zunächst die auf der Diskette vorhandenen Dateien aus. Der Benutzer kann nun mittels Maus oder Cursortasten die Datei wählen, ab der die Wiederherstellung der gesicherten Dateien beginnen soll.
Angenommen, auf der n-ten Backup-Diskette tritt ein Lesefehler auf, so daß die weitere Ausführung abgebrochen wird. Es sei weiterhin angenommen, daß die Ursache für den Lesefehler zwei defekte Spuren sind, die von den Dateien »text1« und »text2« überdeckt werden. SBackup hat also bei »text1« abgebrochen. Wählt man nun Teilrestore und die nächste Datei »text2« als Startdatei, so wird auch diese Aktion abgebrochen, da diese ebenfalls defekt ist. Ab »text3« können jedoch alle folgenden Dateien ohne Einschränkung zurückgeschrieben werden.

- File Restore

Stellt einzelne Dateien einer beliebigen Diskette wieder her. »File Restore« gibt eine Liste der auf der Diskette verfügbaren

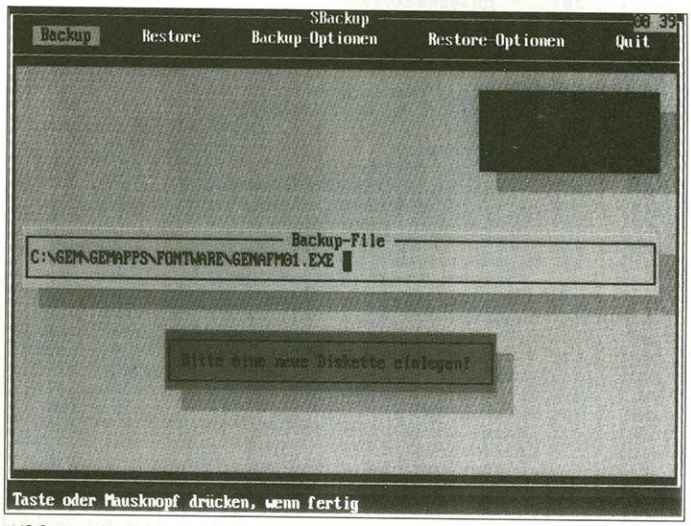


Bild 3. Die komfortable Menü- und Dialogsteuerung ermöglicht ein problemloses Arbeiten

Dateien aus, und der Benutzer kann die wiederherzustellenden Dateien genauso wie bei der Backup-Option »Quellaufwerk« auswählen.

Backup

Der Menüpunkt »Backup« beinhaltet kein Untermenü und startet direkt die Ausführung des Backups (Bild 3). Voraussetzung dafür ist allerdings, daß ein Quellaufwerk definiert ist. Es werden alle gängigen Diskettenformate unterstützt, wobei die Diskette zwar nicht leer, aber formatiert sein muß. Ist die Diskette nicht formatiert oder unlesbar, teilt SBackup dies mit, und die Diskette kann, falls gewünscht, formatiert werden. Die Formatierung auf 720 KByte bei MFM-Laufwerken ist nur durchführbar, falls das BIOS des Computers dies unterstützt oder ein entsprechender Treiber geladen ist. Erkennt das Programm einen Diskettenfehler, wird mit dem Backup abgebrochen. Läßt sich eine Diskette nicht oder nur mit Fehlern formatieren, so wird sie nicht anerkannt.

Allgemeine Fehler

Die Backup-Ausführung wird nicht abgebrochen, wenn sich eine Datei nicht öffnen läßt. Der Benutzer erhält eine Meldung, und es wird mit der nächsten Datei fortgefahren.
SBackup greift auf Informationen aus dem Bootsektor einer Diskette zu. Ist der Bootsektor einer Diskette beschädigt, kann (muß aber nicht unbedingt) eine Unterbrechung auftreten.

Eigenschaften von SBackup

SBackup nutzt fast den gesamten Diskettenplatz aus (keine unnötigen Sektoren für Inhaltsverzeichnis oder ähnliches, nur der Bootsektor wird benötigt).
Außerdem werden verschiedene Diskettenformate innerhalb eines Backups anerkannt. So können zum Beispiel auf einem 1,2-MByte-Laufwerk sowohl 360-KByte- als auch 1,2-MByte-Disketten verwendet werden.
SBackup zeigt sowohl bei Backup als auch bei Restore die jeweils verstrichene Übertragungszeit in einem eigenen Fenster an. Dabei wird nur die reine Übertragungsdauer ermittelt. Zeit, die zum Beispiel für Diskettenwechsel oder beim Formatieren verbraucht wird, geht nicht in die Anzeige ein.

Wichtige Hinweise

In der vorliegenden Version von SBackup muß ein Backup beispielsweise von Laufwerk C auch wieder darauf zurückgespielt werden. Backup- und Restore-Laufwerk müssen somit identisch sein.
Eine fehlerfreie Funktion in Netzwerken kann nicht garantiert werden. Bei einigen BIOS-Versionen von XT und AT kann es vorkommen, daß die Formatierroutine nicht richtig arbeitet. Es sind dies die AT-BIOS-Versionen, die vor dem 15.11.1985, und die XT-BIOS-Versionen, die vor dem 10.01.1986 erzeugt wurden.

(Frank Emminghaus/hs)

```
[K64]    1: { Programm: sbackup.pas
[P35]    2:   Funktion: Multi-Backup-System
[8Z5]    3:   Sprache:  Turbo Pascal ab Version 5.5
[Ø63]    4:             MASM ab Version 4.Ø
[R94]    5:   Autor:    Frank Emminghaus
[P35]    6:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    7:
[9F6]    8: uses windows,dos,dosadd,strings,crt,
[M84]    9:      backwins,maus,diskformat;
[233]   1Ø: {$M 4ØØØØ 1ØØØØØ 655361Ø}
[ØØØ]   11:
[676]   12: function longmul(x,y:word):longint;
[L93]   13: inline($5a/$58/$f7/$e2);
[ØØØ]   14:
[XY4]   15: procedure elpstime; forward;
[RF4]   16: procedure errorh; forward;
[ØØØ]   17:
[MX5]   18: function pseudo(s:string):string;
[MUØ]   19:  begin
[5C6]   2Ø:  if s=p1 then pseudo:=p2 else pseudo:=s
[ZNØ]   21:  end;
[ØØØ]   22:
[KP3]   23: procedure escpressed;
[P12]   24: var ch:char;
[MUØ]   25:  begin
[Y23]   26:  if keypressed then
[MUØ]   27:     begin
[222]   28:     ch:=readkey;
[265]   29:     abbruch:=abbruch or (ord(ch)=27)
[4MØ]   3Ø:     end
[ZNØ]   31:  end;
[ØØØ]   32:
[E53]   33: procedure getboot;
[8U4]   34: var a,q,s,h:byte; b:menufeld;
[MUØ]   35:  begin
[262]   36:  a:=textattr;
[GT2]   37:  sidemenu:=false;
[1R5]   38:  textattr:=col2; normrahmen:=col2;
[NT6]   39:  s:=selectattribut; selectattribut:=sel2;
[3K4]   4Ø:  h:=hotattr; hotattr:=hot3;
[B74]   41:  b[1]:='^36ØK '; b[2]:='1.^2M';
[HZ1]   42:  b[3]:='^72ØK';
[CØ5]   43:  b[4]:='1.^44M'; b[5]:='^Abbruch';
[MX3]   44:  rahmen(2Ø,8,36,9,'Fehler');
[US2]   45:  nomaus; clrscr;
[8N5]   46:  writeln(' Schreib/Lesefehler auf'+
[6F2]   47:           ' Bootsektor!');
[RG6]   48:  writeln(' Bitte die Diskettengröße'+
[UM1]   49:           ' angeben');
[BM1]   5Ø:  showmaus;
[B64]   51:  q:=menumode; menumode:=Ø;
[VC2]   52:  shadow:=false;
[LD2]   53:  vlinewidth:=2;
[GL3]   54:  verticalmenu(33,11,5,b);
[613]   55:  if not escflag then
[MUØ]   56:     begin
[ØR2]   57:     case menumode of
```

```
[351]    58:      1: begin
[2T3]    59:         bootsec.disksize:=72Ø;
[9H3]    6Ø:         bootsec.spursize:=9
[ZNØ]    61:         end;
[551]    62:      2: begin
[VY3]    63:         bootsec.disksize:=24ØØ;
[YH3]    64:         bootsec.spursize:=15
[ZNØ]    65:         end;
[751]    66:      3: begin
[ZY3]    67:         bootsec.disksize:=144Ø;
[9H3]    68:         bootsec.spursize:=9
[ZNØ]    69:         end;
[951]    7Ø:      4: begin
[FZ3]    71:         bootsec.disksize:=288Ø;
[1K3]    72:         bootsec.spursize:=18
[ZNØ]    73:         end;
[UL2]    74:      5: abbruch:=true
[ZNØ]    75:         end;
[7L3]    76:     bootsec.sektsize:=512
[TØ4]    77:     end else abbruch:=true;
[X13]    78:  if not abbruch then
[MUØ]    79:     begin
[M56]    8Ø:     bootsec.vol:='back'+wordstr(diskno);
[FK5]    81:     writesektor(rlw,Ø,1,bootsec,nob)
[ZNØ]    82:     end;
[HD2]    83:  vlinewidth:=1;
[X35]    84:  shadow:=true;  sidemenu:=true;
[MR5]    85:  menumode:=q; ersatz:=not abbruch;
[DU4]    86:  textattr:=a; normrahmen:=a;
[855]    87:  selectattribut:=s; hotattr:=h;
[VZ4]    88:  restorescreen; restorescreen
[ZNØ]    89:  end;
[ØØØ]    9Ø:
[3D6]    91: procedure readboot(var error:boolean);
[KX6]    92: var r:registers; s,d,p:word; count:byte;
[MUØ]    93:  begin
[7G1]    94:  count:=2;
[P31]    95:  repeat
[PK1]    96:  r.ax:=$2Ø1;
[863]    97:  r.bx:=ofs(bootsec);
[663]    98:  r.es:=seg(bootsec);
[W51]    99:  r.cx:=1;
[3M6]   1ØØ:  if restflag then r.dx:=rlw else r.dx:=lw;
[YR1]   1Ø1:  intr($13,r);
[LG3]   1Ø2:  s:=bootsec.sektsize;
[2F3]   1Ø3:  d:=bootsec.disksize;
[TH3]   1Ø4:  p:=bootsec.spursize;
[GX5]   1Ø5:  error:=boolean(r.flags and fcarry)or
[8V5]   1Ø6:  not((s=512)or(s=256)or(s=128)or(s=1Ø24))
[F45]   1Ø7:  or not((d=72Ø)or(d=24ØØ)or(d=144Ø)or
[HE5]   1Ø8:  (d=288Ø)or(d=36Ø)or(d=64Ø)or(d=32Ø))or
[1C5]   1Ø9:  not((p=9)or(p=15)or(p=18)or(p=8));
[FX1]   11Ø:  dec(count);
[CN4]   111:  until not error or (count=Ø)
[ZNØ]   112:  end;
[ØØØ]   113:
[DW6]   114: procedure formatiere(var error:boolean);
```

```
[7K5]  115: var typ,q,a,s,h:byte; b:menufeld;
[XØ2]  116:     label er,er2;
[ØØØ]  117:
[P33]  118:   procedure anzeige;
[MUØ]  119:   begin
[9R5]  12Ø:   textattr:=col4; normrahmen:=col4;
[2W2]  121:   rahmen(3Ø,19,2Ø,1,'');
[DC1]  122:   nomaus;
[DB1]  123:   clrscr;
[NG1]  124:   showmaus
[ZNØ]  125:   end;
[ØØØ]  126:
[MUØ]  127:  begin
[262]  128:  a:=textattr;
[NT6]  129:  s:=selectattribut; selectattribut:=sel2;
[3K4]  13Ø:  h:=hotattr; hotattr:=hot3;
[2G3]  131:  getdrivepar(lw,typ);
[A43]  132:  if typ=1 then begin
[AW3]  133:                 setpar(lw,typ,1,error);
[FE1]  134:                 anzeige;
[Y63]  135:                 format(lw,4Ø,error);
[RA3]  136:                 if error then goto er
[4MØ]  137:                 end
[M23]  138:    else if typ=3 then
[MUØ]  139:         begin
[KW3]  14Ø:         setpar(lw,typ,3,error);
[FE1]  141:         anzeige;
[E73]  142:         format(lw,8Ø,error);
[RA3]  143:         if error then goto er
[4MØ]  144:         end
[K23]  145:     else if typ=2 then
[MUØ]  146:          begin
[484]  147:          b[1]:='^36ØK'; b[2]:='^72ØK';
[3Z1]  148:          b[3]:='^1.2M';
[1R5]  149:          textattr:=col2; normrahmen:=col2;
[Z22]  15Ø:          q:=menumode;
[SZ1]  151:          menumode:=Ø;
[2M3]  152:          verticalmenu(35,15,3,b);
[GH2]  153:          restorescreen;
[613]  154:          if not escflag then
[MUØ]  155:             begin
[FE1]  156:             anzeige;
[ØR2]  157:             case menumode of
[351]  158:             1: begin
[AW3]  159:                setpar(lw,typ,1,error);
[Y63]  16Ø:                format(lw,4Ø,error);
[RA3]  161:                if error then goto er
[ZNØ]  162:                end;
[551]  163:             2: begin
[KW3]  164:                setpar(lw,typ,3,error);
[E73]  165:                format(lw,8Ø,error);
[RA3]  166:                if error then goto er
[ZNØ]  167:                end;
[751]  168:             3: begin
[EW3]  169:                setpar(lw,typ,2,error);
[E73]  17Ø:                format(lw,8Ø,error);
[RA3]  171:                if error then goto er
[4MØ]  172:                end
[4MØ]  173:              end
[ØC2]  174:           end else begin
[352]  175:                    error:=true;
[W32]  176:                    menumode:=q;
[3D1]  177:                    goto er2
[ZNØ]  178:                    end;
[W32]  179:           menumode:=q;
[4MØ]  18Ø:           end
[WP1]  181:         else begin
[9Z1]  182:              b[1]:='^72ØK';
[A62]  183:              b[2]:='^1.44M';
[BR2]  184:              textattr:=col2;
[PY2]  185:              normrahmen:=col2;
[Z22]  186:              q:=menumode;
[SZ1]  187:              menumode:=Ø;
[1M3]  188:              verticalmenu(35,15,2,b);
[GH2]  189:              restorescreen;
[613]  19Ø:              if not escflag then
[MUØ]  191:                 begin
[FE1]  192:                 anzeige;
[ØR2]  193:                 case menumode of
[351]  194:                 1: begin
[KW3]  195:                    setpar(lw,typ,3,error);
[E73]  196:                    format(lw,8Ø,error);
[RA3]  197:                    if error then goto er
[ZNØ]  198:                    end;
[551]  199:                 2: begin
[PW3]  2ØØ:                    setpar(lw,typ,4,error);
[E73]  2Ø1:                    format(lw,8Ø,error);
[RA3]  2Ø2:                    if error then goto er
[4MØ]  2Ø3:                    end
[4MØ]  2Ø4:                 end
[ØC2]  2Ø5:                 end else begin
[352]  2Ø6:                         error:=true;
[W32]  2Ø7:                         menumode:=q;
[3D1]  2Ø8:                         goto er2
[ZNØ]  2Ø9:                         end;
[W32]  21Ø:           menumode:=q;
[ZNØ]  211:           end;
[8LØ]  212:  er:
[NZ4]  213:  restorescreen;    {anzeige}
[E42]  214:  if error then
[MUØ]  215:    begin
```

```
[DTØ]  216:     beep;
[ER2]  217:     textattr:=col3;
[RY2]  218:     normrahmen:=col3;
[LW2]  219:     rahmen(15,19,5Ø,1,'');
[DC1]  22Ø:     nomaus;
[DB1]  221:     clrscr;
[9T4]  222:     write(' Diskette nicht forma'
[PN4]  223:      +'tierbar ! Bitte wechseln.');
[PD4]  224:     infoline(' Taste oder Maus'
[V73]  225:            +'knopf drücken');
[BM1]  226:     showmaus;
[RG4]  227:     repeat until keypressed or
[CA3]  228:          button1 or button2;
[9A4]  229:     if keypressed then clrbuf;
[LY1]  23Ø:     infoline('');
[GH2]  231:     restorescreen;
[AM1]  232:     nobutton;
[ZNØ]  233:     end;
[2MØ]  234:  er2:
[DU4]  235:  textattr:=a; normrahmen:=a;
[855]  236:  selectattribut:=s; hotattr:=h;
[MG1]  237:  nobutton
[ZNØ]  238:  end;
[ØØØ]  239:
[H47]  24Ø: procedure nichtlesbar(var error:boolean);
[137]  241: var a,q,s,h:byte; b:menufeld; r:registers;
[MUØ]  242:  begin
[K33]  243:  a:=textattr; beep;
[GT2]  244:  sidemenu:=false;
[1R5]  245:  textattr:=col2; normrahmen:=col2;
[NT6]  246:  s:=selectattribut; selectattribut:=sel2;
[3K4]  247:  h:=hotattr; hotattr:=hot3;
[WG3]  248:  b[2]:='^formatieren';
[7F3]  249:  b[1]:='^wiederholen';
[U13]  25Ø:  b[3]:='^abbrechen';
[PX3]  251:  rahmen(2Ø,9,36,7,'Fehler');
[US2]  252:  nomaus; clrscr;
[E95]  253:  writeln(' Diskette in Laufwerk ',
[T42]  254:           chr(65+lw),':');
[KB6]  255:  writeln(' kann nicht gelesen werden !');
[CU3]  256:  q:=menumode; showmaus;
[VC2]  257:  shadow:=false;
[LD2]  258:  vlinewidth:=2;
[SZ1]  259:  menumode:=Ø;
[9L3]  26Ø:  verticalmenu(31,12,3,b);
[CP4]  261:  vlinewidth:=1; shadow:=true;
[613]  262:  if not escflag then
[MUØ]  263:     begin
[ØR2]  264:     case menumode of
[BEØ]  265:     1: ;
[AD3]  266:     2: formatiere(error);
[PL2]  267:     3: abbruch:=true
[4MØ]  268:      end
[TØ4]  269:     end else abbruch:=true;
[2W4]  27Ø:  menumode:=q; sidemenu:=true;
[DU4]  271:  textattr:=a; normrahmen:=a;
[855]  272:  selectattribut:=s; hotattr:=h;
[VZ4]  273:  restorescreen; restorescreen
[ZNØ]  274:  end;
[ØØØ]  275:
[N47]  276: procedure wiederholen(var error:boolean);
[AV4]  277: var a:byte; mel:string[33];
[MUØ]  278:  begin
[Y24]  279:  a:=textattr; on:=false;
[DTØ]  28Ø:  beep;
[HS2]  281:  r.ah:=$59; r.bx:=Ø;
[RF1]  282:  msdos(r);
[C56]  283:  if r.ax=$15 then mel:=' Diskettenlauf'+
[VA3]  284:                     'werk nicht bereit !'
[NB5]  285:   else mel:=' Schreib-/Lesefehler'+
[972]  286:              ' (Diskette) !';
[5R5]  287:  textattr:=col3; normrahmen:=col3;
[LW2]  288:  rahmen(15,17,5Ø,2,'');
[US2]  289:  nomaus; clrscr;
[292]  29Ø:  writeln(mel);
[GS5]  291:  write(' <w>iederholen, <a>bbrechen,'+
[CW2]  292:       ' <i>gnorieren ?');
[BM1]  293:  showmaus;
[P31]  294:  repeat
[H31]  295:  ch:='c';
[RG4]  296:  repeat until keypressed or
[CA3]  297:          button1 or button2;
[E65]  298:  if keypressed then ch:=readkey;
[KP3]  299:  if button1 then ch:='w';
[2P5]  3ØØ:  if button2 or (ord(ch)=27) then ch:='a';
[NL5]  3Ø1:  until ch in ['a','A','w','W','I','i'];
[R44]  3Ø2:  abbruch:=(ch in ['a','A']);
[1M3]  3Ø3:  if (ch in ['I','i']) then
[YA2]  3Ø4:   if (r.ax=8) then
[MUØ]  3Ø5:      begin
[5G1]  3Ø6:      getboot;
[U42]  3Ø7:      error:=false
[TØ4]  3Ø8:      end else abbruch:=true;
[344]  3Ø9:  restorescreen; on:=true;
[DU4]  31Ø:  textattr:=a; normrahmen:=a;
[MG1]  311:  nobutton
[ZNØ]  312:  end;
[ØØØ]  313:
```

Listing 1. »sbackup.pas« ist das Turbo-Pascal-5.5-Hauptprogramm des Multi-Backup-Systems »SBackup«

```
[5Z6]  314: function searchfirst(var sekt,ofs:word;
[TC4]  315:          var ende:boolean):boolean;
[4R6]  316: var i,k,maxsekt,sektzahl,aktsekt:word;
[HM5]  317:     found:boolean; p:^recogn; label 1;
[MUØ]  318:  begin
[3A4]  319:  found:=false; ende:=false;
[4W4]  32Ø:  maxsekt:=bootsec.disksize-1;
[LC5]  321:  while (nextsektor<=maxsekt) and
[V22]  322:        not found do
[MUØ]  323:      begin
[3R3]  324:      aktsekt:=nextsektor;
[B35]  325:      if nextsektor+79<=maxsekt then
[LØ2]  326:                        sektzahl:=8Ø
[K96]  327:       else sektzahl:=maxsekt-nextsektor+1;
[P31]  328:      repeat
[896]  329:      readsektor(rlw,nextsektor,sektzahl,
[3U2]  33Ø:                 puffer^,error);
[L24]  331:      if error then sektzahl:=
[PA2]  332:                           sektzahl shr 1
[X95]  333:      until not error or(sektzahl=Ø);
[W63]  334:      if error then begin
[ZB1]  335:                    errorh;
[DZØ]  336:                    goto 1
[ZNØ]  337:                    end;
[TC4]  338:      inc(nextsektor,sektzahl);
[X45]  339:      k:=sektzahl*bootsec.sektsize;
[B73]  34Ø:      p:=pointer(puffer);
[UPØ]  341:      i:=Ø;
[DD4]  342:      while not found and (i<k) do
[MUØ]  343:         begin
[BL2]  344:         if p^[Ø]=228 then
[6M2]  345:          if p^[1]=196 then
[PL2]  346:            if p^[2]=243 then
[1M2]  347:             if p^[3]=171 then
[WL2]  348:              if p^[4]=235 then
[B42]  349:                found:=true;
[951]  35Ø:         inc(i);
[6U2]  351:         inc(ptrec(p).ofs)
[4MØ]  352:         end
[ZNØ]  353:      end;
[Ø42]  354:    if found then
[MUØ]  355:       begin
[DR3]  356:       sekt:=aktsekt+(i-1) div
[923]  357:                        bootsec.sektsize;
[S45]  358:       ofs:=(i-1) mod bootsec.sektsize;
[AL2]  359:       ende:=(p^[4]=92);
[2M3]  36Ø:       ptrec(p).ofs:=i-1-ofs;
[VD4]  361:       if i>bootsec.sektsize then
[E26]  362:          move(p^,puffer^,bootsec.sektsize)
[ZNØ]  363:       end;
[2F3]  364:  searchfirst:=found;
[G8Ø]  365:  1:
[ZNØ]  366:  end;
[ØØØ]  367:
[Z63]  368: procedure lwreset;
[MUØ]  369:  begin
[5C1]  37Ø:  r.ah:=$36;
[ØG4]  371:  if restflag then r.dl:=rlw+1
[5F2]  372:               else r.dl:=lw+1;
[RF1]  373:  msdos(r);
[ZNØ]  374:  end;
[ØØØ]  375:
[D13]  376: procedure smooth;
[VØ6]  377: var spuren:integer; error:boolean;
[W55]  378:     s: array[1..512] of byte; k:byte;
[MUØ]  379:  begin
[ØZ5]  38Ø:  if restflag then k:=rlw else k:=lw;
[7G4]  381:  spuren:=bootsec.disksize-
[ØE3]  382:           8*bootsec.spursize;
[CR2]  383:  while spuren>Ø do
[MUØ]  384:        begin
[CB5]  385:        readsektor(k,spuren,1,s,error);
[PD5]  386:        dec(spuren,8*bootsec.spursize);
[4MØ]  387:        end
[ZNØ]  388:  end;
[ØØØ]  389:
[9V3]  39Ø: procedure backmeldung;
[MUØ]  391:  begin
[9R5]  392:  textattr:=col4; normrahmen:=col4;
[XL4]  393:  rahmen(3,12,73,1,'Backup-File');
[DC1]  394:  nomaus;
[DB1]  395:  clrscr;
[NG1]  396:  showmaus
[ZNØ]  397:  end;
[ØØØ]  398:
[8Z3]  399: procedure restmeldung;
[MUØ]  4ØØ:  begin
[9R5]  4Ø1:  textattr:=col4; normrahmen:=col4;
[ØW4]  4Ø2:  rahmen(3,12,73,1,'Restore-File');
[DC1]  4Ø3:  nomaus;
[DB1]  4Ø4:  clrscr;
[NG1]  4Ø5:  showmaus
[ZNØ]  4Ø6:  end;
[ØØØ]  4Ø7:
[CU4]  4Ø8: function falsedisk:boolean;
[GB5]  4Ø9: var a:byte; w:word; er:boolean;
[T35]  41Ø:     mel:string[6Ø]; b:string[8];
[TH3]  411:     c:string[8]; q:byte;
[MUØ]  412:  begin
[262]  413:  a:=textattr;
```

```
[MX3]  414:  q:=length(bootsec.vol);
[5X3]  415:  b:=right(bootsec.vol,4);
[YZ2]  416:  b:=scanright(b,'k');
[WX3]  417:  c:=left(bootsec.vol,4);
[AS3]  418:  w:=wordval(b,er); beep;
[G64]  419:  if (w=Ø)or(c<>'back')or(q>1Ø)
[3B6]  42Ø:   then mel:=' Diskette im Laufwerk ist'+
[H84]  421:                ' keine Backup-Diskette ! '
[DB6]  422:   else mel:=' Falsche Diskette (Backup-'+
[9A5]  423:        'Diskette Nr. '+b+') im Laufwerk !';
[5R5]  424:  textattr:=col3; normrahmen:=col3;
[6S2]  425:  rahmen(8,18,63,2,'');
[US2]  426:  nomaus; clrscr;
[RD5]  427:  infoline(' Taste oder Mausknopf'+
[EV1]  428:                ' drücken');
[292]  429:  writeln(mel);
[9G4]  43Ø:  write(' <w>iederholen oder'+
[NK2]  431:         ' <a>bbrechen ?');
[BM1]  432:  showmaus;
[P31]  433:  repeat
[H31]  434:  ch:='c';
[RG4]  435:  repeat until keypressed or
[CA3]  436:         button1 or button2;
[E65]  437:  if keypressed then ch:=readkey;
[KP3]  438:  if button1 then ch:='w';
[2P5]  439:  if button2 or (ord(ch)=27) then ch:='a';
[5K4]  44Ø:  until ch in ['a','A','w','W'];
[HU4]  441:  falsedisk:=not(ch in ['w','W']);
[LY1]  442:  infoline('');
[GH2]  443:  restorescreen;
[DU4]  444:  textattr:=a; normrahmen:=a;
[MG1]  445:  nobutton
[ZNØ]  446:  end;
[ØØØ]  447:
[SZ2]  448: procedure errorh;
[SG5]  449: var a:byte; ch:char; r:registers;
[ØW2]  45Ø:     mel:string[62];
[MUØ]  451:  begin
[UN1]  452:  on:=false;
[DTØ]  453:  beep;
[HS2]  454:  r.ah:=$59; r.bx:=Ø;
[RF1]  455:  msdos(r);
[6H2]  456:  if not fehl then
[FT5]  457:     mel:=' Schreib-/Lesefehler auf Disk'+
[AZ4]  458:          'ette! Aktion wird abgebrochen'
[ZRØ]  459:   else
[MUØ]  46Ø:    begin
[VW5]  461:    if r.bh=1 then mel:=' Kein Platz mehr'+
[MG4]  462:              ' auf Ziellaufwerk! Aktion'+
[813]  463:              ' wird abgebrochen'
[PU5]  464:     else if r.bh=3 then mel:=' Datei exi'+
[6E4]  465:          'stiert bereits und ist'+
[NK3]  466:          ' schreibgeschützt!'
[E16]  467:      else mel:=' Schreib-/Lesefehler auf'+
[ED4]  468:           ' Festplatte! Aktion wird'+
[DA2]  469:           ' abgebrochen!';
[ZNØ]  47Ø:    end;
[262]  471:  a:=textattr;
[9S4]  472:  abbruch:=not fehl or(r.bh<>3);
[5R5]  473:  textattr:=col3; normrahmen:=col3;
[3S2]  474:  rahmen(9,18,62,1,'');
[DC1]  475:  nomaus;
[DB1]  476:  clrscr;
[PW1]  477:  write(mel);
[RD5]  478:  infoline(' Taste oder Mausknopf'+
[EV1]  479:                ' drücken');
[BM1]  48Ø:  showmaus;
[RG4]  481:  repeat until keypressed or
[CA3]  482:         button1 or button2;
[E65]  483:  if keypressed then ch:=readkey;
[VE4]  484:  if (ord(ch)=27)and not abbruch
[SA3]  485:     then abbruch:=true;
[583]  486:  infoline(''); clrbuf;
[T96]  487:  restorescreen; fehl:=false; on:=true;
[DU4]  488:  textattr:=a; normrahmen:=a;
[MG1]  489:  nobutton
[ZNØ]  49Ø:  end;
[ØØØ]  491:
[ZB4]  492: procedure verzeicherror;
[AA3]  493: var a:byte; ch:char;
[MUØ]  494:  begin
[GK2]  495:  on:=false; beep;
[262]  496:  a:=textattr;
[5R5]  497:  textattr:=col3; normrahmen:=col3;
[3S2]  498:  rahmen(9,18,62,1,'');
[US2]  499:  nomaus; clrscr;
[NH6]  5ØØ:  write(' Verzeichnisfehler! Aktion wird'+
[7B2]  5Ø1:         ' abgebrochen');
[RD5]  5Ø2:  infoline(' Taste oder Mausknopf'+
[EV1]  5Ø3:                ' drücken');
[BM1]  5Ø4:  showmaus;
[RG4]  5Ø5:  repeat until keypressed or
[CA3]  5Ø6:         button1 or button2;
[9A4]  5Ø7:  if keypressed then clrbuf;
[2G4]  5Ø8:  infoline(''); abbruch:=true;
[T96]  5Ø9:  restorescreen; fehl:=false; on:=true;
[DU4]  51Ø:  textattr:=a; normrahmen:=a;
[MG1]  511:  nobutton
[ZNØ]  512:  end;
[ØØØ]  513:
```

```
[TY2]  514: procedure fertig;
[AA3]  515: var a:byte; ch:char;
[MUØ]  516:  begin
[262]  517:  a:=textattr;
[1R5]  518:  textattr:=col2; normrahmen:=col2;
[DC1]  519:  nomaus;
[DW2]  52Ø:  rahmen(35,11,12,3,'');
[DB1]  521:  clrscr;
[NV1]  522:  gotoxy(3,2);
[5K2]  523:  write('Fertig!');
[BM1]  524:  showmaus;
[RD5]  525:  infoline(' Taste oder Mausknopf'+
[EV1]  526:                ' drücken');
[RG4]  527:  repeat until keypressed or
[CA3]  528:         button1 or button2;
[9A4]  529:  if keypressed then clrbuf;
[DU4]  53Ø:  textattr:=a; normrahmen:=a;
[LY1]  531:  infoline('');
[GH2]  532:  restorescreen;
[MG1]  533:  nobutton
[ZNØ]  534:  end;
[ØØØ]  535:
[KX6]  536: function searchsektor(anf:word; var ps:
[ZB5]  537:          word;var ende:boolean):boolean;
[A76]  538: var found:boolean; i:word; p:^recogn;
[MUØ]  539:  begin
[986]  54Ø:  p:=pointer(puffer); ptrec(p).ofs:=anf;
[7F3]  541:  found:=false; i:=anf;
[P75]  542:  while (i<= bootsec.sektsize-6)and
[V22]  543:        not found do
[MUØ]  544:        begin
[BL2]  545:        if p^[Ø]=228 then
[6M2]  546:         if p^[1]=196 then
[PL2]  547:          if p^[2]=243 then
[1M2]  548:           if p^[3]=171 then
[FV4]  549:            if p^[4]=235 then found:=true;
[951]  55Ø:        inc(i);
[6U2]  551:        inc(ptrec(p).ofs)
[ZNØ]  552:        end;
[AB4]  553:  ende:=(p^[4]=92) and found;
[P61]  554:  ps:=i-1;
[GG3]  555:  searchsektor:=found
[ZNØ]  556:  end;
[ØØØ]  557:
[E15]  558: procedure nextsekt; forward;
[PØ5]  559: procedure nextdisk1; forward;
[ØØØ]  56Ø:
[W66]  561: procedure scandisk(var ende:boolean);
[8B5]  562: type prec=record size:longint;
[DU1]  563:                  attr:byte;
[L62]  564:                  time:longint
[ZNØ]  565:                  end;
[ØØØ]  566:
[V46]  567: var p:prec; s:string; size:longint;
[C83]  568:     anf,i,oldi,ps:word;
[894]  569:     puf1:pointer; label err1;
[MUØ]  57Ø:  begin
[1H3]  571:  dfsize:=Ø; oldi:=6ØØØØ;
[272]  572:  puf1:=puffer;
[VG6]  573:  inc(ptrec(puf1).ofs,bootsec.sektsize);
[YØ5]  574:  if searchfirst(i,anf,ende) then
[MUØ]  575:     begin
[B64]  576:     if abbruch then goto err1;
[V94]  577:     while (i<bootsec.disksize)
[6N2]  578:           and not ende do
[MUØ]  579:         begin
[ND2]  58Ø:         if i<>oldi then
[2L4]  581:            reads(rlw,i,1,puffer,error);
[W63]  582:         if error then begin
[ZB1]  583:                       errorh;
[5H1]  584:                       goto err1
[ZNØ]  585:                       end;
[WG5]  586:         if searchsektor(anf,ps,ende) then
[FG2]  587:          if not ende then
[MUØ]  588:             begin
[ZP2]  589:             nextsektor:=i+1;
[Z24]  59Ø:             if i<>oldi then nextsekt;
[B64]  591:             if abbruch then goto err1;
[6E2]  592:             if (i<>oldi)and
[HG4]  593:                (i=bootsec.disksize-1) then
[MUØ]  594:                begin
[881]  595:                dec(i,2);
[VL1]  596:                nextdisk1
[ZNØ]  597:                end;
[B64]  598:             if abbruch then goto err1;
[1Ø2]  599:             inc(dfsize);
[ZB4]  6ØØ:             sektfeld[dfsize].sekt:=i;
[AD4]  6Ø1:             sektfeld[dfsize].ofs:=ps;
[HH1]  6Ø2:             inc(ps,5);
[FK3]  6Ø3:             move(puffer^[ps],s,
[HU2]  6Ø4:                 puffer^[ps]+1);
[TT3]  6Ø5:             inc(ps,puffer^[ps]+1);
[2W3]  6Ø6:             move(puffer^[ps],p,9);
[NH1]  6Ø7:             inc(ps,9);
[C84]  6Ø8:             dateifeld[dfsize].name:=
[TP2]  6Ø9:                 scanright(s,'\');
[2A4]  61Ø:             dateifeld[dfsize].size:=
[E91]  611:                  p.size;
[C94]  612:             dateifeld[dfsize].time:=
[S81]  613:                  p.time;
[Ø94]  614:             dateifeld[dfsize].attr:=
[C81]  615:                  p.attr;
[BB2]  616:             size:=p.size;
[US4]  617:             if ps>=bootsec.sektsize then
[MUØ]  618:                begin
[951]  619:                inc(i);
[D44]  62Ø:                dec(ps,bootsec.sektsize)
[ZNØ]  621:                end;
[T15]  622:             if size>bootsec.sektsize then
[MUØ]  623:                begin
[YH4]  624:                dec(size,bootsec.sektsize-
[SLØ]  625:                    ps);
[P92]  626:                inc(i,size div
[P63]  627:                    bootsec.sektsize);
[452]  628:                anf:=size mod
[923]  629:                     bootsec.sektsize;
[K54]  63Ø:                if anf>bootsec.sektsize-5
[ZN1]  631:                   then begin
[951]  632:                        inc(i);
[8YØ]  633:                        anf:=Ø
[4MØ]  634:                        end
[4MØ]  635:             end
[WP1]  636:            else begin
[Z42]  637:                 inc(ps,size);
[441]  638:                 dec(i);
[SØ4]  639:                 if ps>=bootsec.sektsize
[ZN1]  64Ø:                    then begin
[951]  641:                         inc(i);
[FK2]  642:                         dec(ps,bootsec.
[NL1]  643:                             sektsize)
[ZNØ]  644:                         end;
[3G1]  645:                 anf:=ps-1
[4MØ]  646:                 end
[ZNØ]  647:          end;
[XXØ]  648:       inc(i)
[ZNØ]  649:    end;
[4W2]  65Ø:    if ende then begin
[1Ø2]  651:                 inc(dfsize);
[ZB4]  652:                 sektfeld[dfsize].sekt:=i;
[M94]  653:                 sektfeld[dfsize].ofs:=ps
[ZNØ]  654:                 end;
[ZNØ]  655:  end;
[DVØ]  656:  err1:
[ZNØ]  657:  end;
[ØØØ]  658:
[9Y6]  659: procedure insdisk;     {nur für backup}
[9Ø6]  66Ø: var ch:char; error:boolean; a:byte;
[Z41]  661:     label 1;
[MUØ]  662:  begin
[262]  663:  a:=textattr;
[1R5]  664:  textattr:=col2; normrahmen:=col2;
[GW2]  665:  rahmen(23,17,32,1,'');
[DC1]  666:  nomaus;
[DB1]  667:  clrscr;
[7B6]  668:  write(' Bitte eine Diskette einlegen!');
[X47]  669:  infoline(' Taste oder Mausknopf drücken'+
[EG2]  67Ø:            ', wenn fertig');
[BM1]  671:  showmaus;
[RG4]  672:  repeat until keypressed or
[CA3]  673:         button1 or button2;
[E65]  674:  if keypressed then ch:=readkey;
[GH2]  675:  restorescreen;
[WS4]  676:  if (ord(ch)=27) or button2 then
[MUØ]  677:     begin
[XD2]  678:     abbruch:=true;
[DZØ]  679:     goto 1
[ZNØ]  68Ø:     end;
[WX2]  681:  nobutton; clrbuf;
[P31]  682:  repeat
[P31]  683:   repeat
[B31]  684:   ch:='a';
[8R2]  685:   readboot(error);
[TD5]  686:   if error then nichtlesbar(error)
[TM4]  687:   until not error or abbruch;
[EP3]  688:  if abbruch then goto 1;
[AH1]  689:  lwreset;
[M56]  69Ø:  bootsec.vol:='back'+wordstr(diskno);
[YR5]  691:  writesektor(lw,Ø,1,bootsec,error);
[XD5]  692:  if error then wiederholen(error)
[TM4]  693:  until not error or abbruch;
[LY1]  694:  infoline('');
[ØF2]  695:  nextsektor:=1;
[XH5]  696:  freesektor:=bootsec.disksize-2;
[3Z4]  697:  freebyte:=bootsec.sektsize;
[3R2]  698:  zuschreiben:=1;
[2Y2]  699:  sektpoi:=puffer;
[G8Ø]  7ØØ:  1:
[DU4]  7Ø1:  textattr:=a; normrahmen:=a;
[MG1]  7Ø2:  nobutton
[ZNØ]  7Ø3:  end;
[ØØØ]  7Ø4:
[1U6]  7Ø5: procedure insdisk1;  {nur für restore}
[886]  7Ø6: var ch:char; error,fd:boolean; a:byte;
[6G3]  7Ø7:     w:string[8]; label 1;
[MUØ]  7Ø8:  begin
[262]  7Ø9:  a:=textattr;
[1R5]  71Ø:  textattr:=col2; normrahmen:=col2;
[GW2]  711:  rahmen(23,17,32,1,'');
```

Listing 1. »sbackup.pas« ist das Turbo-Pascal-5.5-Hauptprogramm des Multi-Backup-Systems »SBackup«

```
[DC1]  712:  nomaus;
[2W3]  713:  clrscr; ersatz:=false;
[7B6]  714:  write(' Bitte eine Diskette einlegen!');
[RD5]  715:  infoline(' Taste oder Mausknopf'+
[UY3]  716:             ' drücken, wenn fertig');
[BM1]  717:  showmaus;
[RG4]  718:  repeat until keypressed or
[CA3]  719:         button1 or button2;
[GH2]  720:  restorescreen;
[E65]  721:  if keypressed then ch:=readkey;
[WS4]  722:  if (ord(ch)=27) or button2 then
[MUØ]  723:     begin
[XD2]  724:     abbruch:=true;
[DZØ]  725:     goto 1
[ZNØ]  726:     end;
[WX2]  727:  nobutton; clrbuf;
[P31]  728:  repeat
[B31]  729:  ch:='a';
[8R2]  730:  readboot(error);
[4Z3]  731:  w:=left(bootsec.vol,4);
[LH5]  732:  if error then wiederholen(error);
[MZ4]  733:  fd:=(w<>'back') and not abbruch
[8V4]  734:     and not error and not ersatz;
[YL4]  735:  if fd then abbruch:=falsedisk
[LF5]  736:  until not(error or fd) or abbruch;
[7M4]  737:  if not abbruch then lwreset;
[LY1]  738:  infoline('');
[ØF2]  739:  nextsektor:=1;
[XH5]  740:  freesektor:=bootsec.disksize-2;
[3Z4]  741:  freebyte:=bootsec.sektsize;
[3R2]  742:  zuschreiben:=1;
[2Y2]  743:  sektpoi:=puffer;
[G8Ø]  744:  1:
[DU4]  745:  textattr:=a; normrahmen:=a;
[MG1]  746:  nobutton
[ZNØ]  747:  end;
[ØØØ]  748:
[XH6]  749: procedure neudisk;  {nur für backup}
[9Ø6]  750: var ch:char; error:boolean; a:byte;
[Z41]  751:     label 1;
[MUØ]  752:  begin
[KØ2]  753:  inc(diskno);
[Y23]  754:  smooth; on:=false;
[262]  755:  a:=textattr;
[1R5]  756:  textattr:=col2; normrahmen:=col2;
[YW2]  757:  rahmen(2Ø,17,37,1,'');
[DC1]  758:  nomaus;
[DB1]  759:  clrscr;
[YF5]  760:  write(' Bitte eine neue Diskette'+
[8Ø2]  761:          ' einlegen!');
[RD5]  762:  infoline(' Taste oder Mausknopf'+
[UY3]  763:             ' drücken, wenn fertig');
[BM1]  764:  showmaus;
[RG4]  765:  repeat until keypressed or
[CA3]  766:         button1 or button2;
[GH2]  767:  restorescreen;
[E65]  768:  if keypressed then ch:=readkey;
[WS4]  769:  if (ord(ch)=27) or button2 then
[MUØ]  770:     begin
[XD2]  771:     abbruch:=true;
[DZØ]  772:     goto 1
[ZNØ]  773:     end;
[WX2]  774:  nobutton; clrbuf;
[P31]  775:  repeat
[P31]  776:   repeat
[B31]  777:   ch:='a';
[8R2]  778:   readboot(error);
[TD5]  779:   if error then nichtlesbar(error)
[TM4]  780:   until not error or abbruch;
[EP3]  781:  if abbruch then goto 1;
[AH1]  782:  lwreset;
[M56]  783:  bootsec.vol:='back'+wordstr(diskno);
[YR5]  784:  writesektor(lw,Ø,1,bootsec,error);
[XD5]  785:  if error then wiederholen(error)
[TM4]  786:  until not error or abbruch;
[LY1]  787:  infoline('');
[ØF2]  788:  nextsektor:=1;
[XH5]  789:  freesektor:=bootsec.disksize-2;
[G8Ø]  790:  1:
[UK1]  791:  on:=true;
[DU4]  792:  textattr:=a; normrahmen:=a;
[MG1]  793:  nobutton
[ZNØ]  794:  end;
[ØØØ]  795:
[BX2]  796: procedure nexts;
[EF4]  797: var error:boolean; label 1;
[MUØ]  798:  begin
[1B6]  799:  writes(lw,nextsektor,1,puffer,error);
[W63]  800:  if error then begin
[ZB1]  801:                errorh;
[DZØ]  802:                goto 1
[ZNØ]  803:                end;
[EP2]  804:  dec(freesektor);
[RT2]  805:  inc(nextsektor);
[BT4]  806:  if freesektor<Ø then neudisk;
[EP3]  807:  if abbruch then goto 1;
[3Z4]  808:  freebyte:=bootsec.sektsize;
[2Y2]  809:  sektpoi:=puffer;
[G8Ø]  810:  1:
[ZNØ]  811:  end;
```

```
[ØØØ]  812:
[RØ3]  813: procedure nexts1;
[FB6]  814: var error:boolean; p:pointer; label 1;
[MUØ]  815:  begin
[1B6]  816:  writes(lw,nextsektor,1,puffer,error);
[W63]  817:  if error then begin
[ZB1]  818:                errorh;
[DZØ]  819:                goto 1
[ZNØ]  820:                end;
[EP2]  821:  dec(freesektor);
[RT2]  822:  inc(nextsektor);
[ØP1]  823:  p:=puffer;
[T26]  824:  inc(ptrec(p).ofs,bootsec.sektsize);
[Z76]  825:  move(p^,puffer^,bootsec.sektsize);
[WY6]  826:  dec(ptrec(sektpoi).ofs,bootsec.sektsize);
[BT4]  827:  if freesektor<Ø then neudisk;
[EP3]  828:  if abbruch then goto 1;
[MG5]  829:  inc(freebyte,bootsec.sektsize);
[G8Ø]  830:  1:
[ZNØ]  831:  end;
[ØØØ]  832:
[8N5]  833: procedure uebertrage(r:storerec;
[1A2]  834:                      path:string);
[8B5]  835: type prec=record size:longint;
[DU1]  836:                  attr:byte;
[L62]  837:                  time:longint
[ZNØ]  838:                  end;
[ØØØ]  839:
[486]  840: var s:string; p1:prec; size:longint;
[FV3]  841:     p:pointer; a,b,c,l:word;
[FE2]  842:     label err1,err2;
[ØØØ]  843:
[MUØ]  844:  begin
[7E5]  845:  p1.size:=r.size; p1.time:=r.time;
[2R2]  846:  p1.attr:=r.attr;
[643]  847:  s:=path+'\'+r.name;
[ER3]  848:  qdatei:=open(s,Ø,fehl);
[VW2]  849:  if fehl then begin
[ZB1]  850:               errorh;
[7H1]  851:               goto err2
[ZNØ]  852:               end;
[DC1]  853:  nomaus;
[XM3]  854:  clrscr; error:=false;
[9G1]  855:  write(s);
[BM1]  856:  showmaus;
[Ø24]  857:  if freebyte<5 then nexts;
[D64]  858:  if abbruch then goto err2;
[1K3]  859:  move(rec,sektpoi^,5);
[A84]  860:  inc(ptrec(sektpoi).ofs,5);
[ØL2]  861:  dec(freebyte,5);
[WD2]  862:  l:=length(s)+1;
[4X3]  863:  move(s[Ø],sektpoi^,l);
[194]  864:  inc(ptrec(sektpoi).ofs,l);
[5N2]  865:  dec(freebyte,l);
[L64]  866:  if freebyte<Ø then nexts1;
[D64]  867:  if abbruch then goto err2;
[U93]  868:  move(p1,sektpoi^,9);
[CL2]  869:  dec(freebyte,9);
[E84]  870:  inc(ptrec(sektpoi).ofs,9);
[AE4]  871:  if freebyte<=Ø then nexts1;
[D64]  872:  if abbruch then goto err2;
[LB2]  873:  size:=r.size;
[ØØØ]  874:
[8L2]  875:  while size>Ø do
[MUØ]  876:    begin
[876]  877:    nochfrei:=freebyte+bootsec.sektsize*
[G22]  878:              freesektor;
[WA4]  879:    if size<$faØØ+freebyte then
[MUØ]  880:       begin
[VN3]  881:       if nochfrei>=size then
[MUØ]  882:          begin
[N55]  883:          c:=readh(qdatei,size,sektpoi^,
[AYØ]  884:                   fehl);
[AR3]  885:          if fehl then goto err1;
[5L3]  886:          if size>freebyte then
[MUØ]  887:             begin
[9B3]  888:             dec(size,freebyte);
[F35]  889:             a:=size div bootsec.sektsize;
[T44]  890:             b:=ptrec(sektpoi).ofs div
[923]  891:                bootsec.sektsize;
[KE4]  892:             b:=(b+a+1)*bootsec.sektsize;
[ØP1]  893:             p:=puffer;
[H73]  894:             inc(ptrec(p).ofs,b);
[NB3]  895:             inc(zuschreiben,a);
[FR3]  896:             writes(lw,nextsektor,
[TL4]  897:               zuschreiben,puffer,error);
[CU3]  898:             if error then goto err1;
[913]  899:             move(p^,puffer^,
[P63]  900:                  bootsec.sektsize);
[D25]  901:             b:=size mod bootsec.sektsize;
[335]  902:             freebyte:=bootsec.sektsize-b;
[2Y2]  903:             sektpoi:=puffer;
[R84]  904:             inc(ptrec(sektpoi).ofs,b);
[5W4]  905:             inc(nextsektor,zuschreiben);
[853]  906:             dec(freesektor,a+1);
[893]  907:             if freesektor<Ø then
[MUØ]  908:                begin
[NF1]  909:                neudisk;
[B64]  910:                if abbruch then goto err1;
[2Y2]  911:                sektpoi:=puffer;
[RR4]  912:                freebyte:=bootsec.sektsize
```

```
[ZNØ]  913:                end;
[3R2]  914:              zuschreiben:=1;
[T81]  915:              size:=Ø
[4MØ]  916:              end
[WP1]  917:            else begin
[AA3]  918:                 dec(freebyte,size);
[KY3]  919:                 inc(ptrec(sektpoi).ofs,
[H11]  92Ø:                     size);
[MA1]  921:                 size:=Ø;
[4MØ]  922:                 end
[4MØ]  923:          end
[ØØØ]  924:
[ZRØ]  925:        else
[M74]  926:         begin   {nochfrei<size}
[N55]  927:          c:=readh(qdatei,size,sektpoi^,
[AYØ]  928:                   fehl);
[AR3]  929:          if fehl then goto err1;
[B25]  93Ø:          dec(size,freebyte+freesektor*
[P63]  931:              bootsec.sektsize);
[T44]  932:          b:=ptrec(sektpoi).ofs div
[923]  933:              bootsec.sektsize;
[M33]  934:          b:=(b+1+freesektor)*
[923]  935:              bootsec.sektsize;
[CP3]  936:          ptrec(sektpoi).ofs:=b;
[ZW5]  937:          writes(lw,nextsektor,zuschreiben+
[3A4]  938:                 freesektor,puffer,error);
[CU3]  939:          if error then goto err1;
[NF1]  94Ø:          neudisk;
[B64]  941:          if abbruch then goto err1;
[F35]  942:          a:=size div bootsec.sektsize;
[9F5]  943:          writes(lw,nextsektor,a,sektpoi,
[291]  944:                 error);
[CU3]  945:          if error then goto err1;
[734]  946:          inc(ptrec(sektpoi).ofs,a*
[P63]  947:              bootsec.sektsize);
[3Z2]  948:          dec(freesektor,a);
[C23]  949:          inc(nextsektor,a);
[3R2]  95Ø:          zuschreiben:=1;
[2Y3]  951:          move(sektpoi^,puffer^,
[P63]  952:               bootsec.sektsize);
[2Y2]  953:          sektpoi:=puffer;
[D25]  954:          b:=size mod bootsec.sektsize;
[XE4]  955:          inc(ptrec(sektpoi).ofs, b);
[335]  956:          freebyte:=bootsec.sektsize-b;
[T81]  957:          size:=Ø
[4MØ]  958:          end
[4MØ]  959:        end
[ØØØ]  96Ø:
[ZRØ]  961:    else
[G85]  962:      begin    {size>=$faØØ+freebyte}
[HT3]  963:      if nochfrei>= size then
[MUØ]  964:        begin
[AY3]  965:        dec(size,freebyte+$faØØ);
[MZ4]  966:        c:=readh(qdatei,$faØØ+freebyte,
[9M2]  967:                 sektpoi^,fehl);
[AR3]  968:        if fehl then goto err1;
[4X4]  969:        a:=$faØØ div bootsec.sektsize;
[NB3]  97Ø:        inc(zuschreiben,a);
[853]  971:        dec(freesektor,a+1);
[1X5]  972:        writes(lw,nextsektor,zuschreiben,
[XC2]  973:               puffer,error);
[CU3]  974:        if error then goto err1;
[5W4]  975:        inc(nextsektor,zuschreiben);
[BT4]  976:        if freesektor<Ø then neudisk;
[B64]  977:        if abbruch then goto err1;
[2Y2]  978:        sektpoi:=puffer;
[3R2]  979:        zuschreiben:=1;
[3Z4]  98Ø:        freebyte:=bootsec.sektsize;
[4MØ]  981:        end
[ZRØ]  982:      else
[MUØ]  983:        begin
[4X4]  984:        a:=$faØØ div bootsec.sektsize;
[1E3]  985:        if a>freesektor then
[MUØ]  986:           begin
[KB2]  987:           b:=freesektor*
[923]  988:              bootsec.sektsize;
[5K4]  989:           c:=readh(qdatei,freebyte+b,
[9M2]  99Ø:              sektpoi^,fehl);
[AR3]  991:           if fehl then goto err1;
[AH3]  992:           dec(size,freebyte+b);
[FR3]  993:           writes(lw,nextsektor,
[F74]  994:              zuschreiben+freesektor,
[XC2]  995:              puffer,error);
[CU3]  996:           if error then goto err1;
[NF1]  997:           neudisk;
[B64]  998:           if abbruch then goto err1;
[3R2]  999:           zuschreiben:=1;
[3Z4] 1ØØØ:           freebyte:=bootsec.sektsize;
[2Y2] 1ØØ1:           sektpoi:=puffer;
[4MØ] 1ØØ2:           end
[ZRØ] 1ØØ3:         else
[MUØ] 1ØØ4:           begin
[F64] 1ØØ5:           c:=readh(qdatei,freebyte+
[RD3] 1ØØ6:                    $faØØ,sektpoi^,fehl);
[AR3] 1ØØ7:           if fehl then goto err1;
[AY3] 1ØØ8:           dec(size,freebyte+$faØØ);
[FR3] 1ØØ9:           writes(lw,nextsektor,
[EV4] 1Ø1Ø:             zuschreiben+a,puffer,error);
[CU3] 1Ø11:           if error then goto err1;
[853] 1Ø12:           dec(freesektor,a+1);
[W75] 1Ø13:           inc(nextsektor,a+zuschreiben);
```

```
[BT4] 1Ø14:           if freesektor<Ø then neudisk;
[B64] 1Ø15:           if abbruch then goto err1;
[2Y2] 1Ø16:           sektpoi:=puffer;
[3R2] 1Ø17:           zuschreiben:=1;
[3Z4] 1Ø18:           freebyte:=bootsec.sektsize;
[4MØ] 1Ø19:           end
[4MØ] 1Ø2Ø:         end
[4MØ] 1Ø21:       end
[ZNØ] 1Ø22:    end;
[ØØØ] 1Ø23:
[DVØ] 1Ø24:   err1:
[7U4] 1Ø25:   if error or fehl then errorh;
[ØH3] 1Ø26:   closeh(qdatei,error);
[EVØ] 1Ø27:   err2:
[ZNØ] 1Ø28:  end;
[ØØØ] 1Ø29:
[ZR5] 1Ø3Ø: procedure backupdir(path:string;
[HB3] 1Ø31:                     var error:boolean);
[GR6] 1Ø32: var dirfeld: array[1..1ØØ] of direintrag;
[HE4] 1Ø33:       dirsize,anzahl,i,j,k:word;
[3V2] 1Ø34:       b,isopen:boolean;
[B12] 1Ø35:       r:storerec;
[ØB2] 1Ø36:       rest:longint;
[7T4] 1Ø37:       u:string[12Ø]; v:string[14];
[CU1] 1Ø38:       label abbr;
[ØØØ] 1Ø39:
[ZX5] 1Ø4Ø:   function infeld(s:string):boolean;
[K85] 1Ø41:   var schl:searchrec; attr:word;
[CW2] 1Ø42:       p,q:string[15Ø];
[MUØ] 1Ø43:    begin
[2K3] 1Ø44:    dfsize:=Ø; dirsize:=Ø;
[F33] 1Ø45:    p:=scanright(s,'\');
[VY3] 1Ø46:    if sflag and (p<>'*.*') then
[MUØ] 1Ø47:       begin
[2D4] 1Ø48:       q:=s; delright(q,length(p));
[BF1] 1Ø49:       q:=q+'*.*';
[ZG4] 1Ø5Ø:       findfirst(q,anyfile,schl);
[EA3] 1Ø51:       while doserror=Ø do
[MUØ] 1Ø52:         begin
[A93] 1Ø53:         if schl.attr=$1Ø then
[T73] 1Ø54:          if (schl.name<>'.')and
[4A3] 1Ø55:             (schl.name<>'..') then
[MUØ] 1Ø56:             begin
[182] 1Ø57:             inc(dirsize);
[AW4] 1Ø58:             dirfeld[dirsize]:=schl.name
[ZNØ] 1Ø59:             end;
[KD2] 1Ø6Ø:         findnext(schl)
[4MØ] 1Ø61:         end
[ZNØ] 1Ø62:       end;
[7H4] 1Ø63:    findfirst(s,anyfile,schl);
[EA3] 1Ø64:    while doserror=Ø do
[MUØ] 1Ø65:       begin
[A93] 1Ø66:       if schl.attr=$1Ø then
[MUØ] 1Ø67:          begin
[T73] 1Ø68:          if (schl.name<>'.')and
[4A3] 1Ø69:             (schl.name<>'..') then
[MUØ] 1Ø7Ø:             begin
[182] 1Ø71:             inc(dirsize);
[AW4] 1Ø72:             dirfeld[dirsize]:=schl.name
[4MØ] 1Ø73:             end
[4MØ] 1Ø74:          end
[ZRØ] 1Ø75:        else
[793] 1Ø76:          if schl.attr<>8 then
[N74] 1Ø77:           if (schl.attr and 6)<>6 then
[MUØ] 1Ø78:              begin
[YW2] 1Ø79:              if not timeflag or
[6R3] 1Ø8Ø:                 (schl.time>searchtime)
[2RØ] 1Ø81:                 then
[MUØ] 1Ø82:                 begin
[1Ø2] 1Ø83:                 inc(dfsize);
[1B4] 1Ø84:                 with dateifeld[dfsize] do
[MUØ] 1Ø85:                      begin
[4P2] 1Ø86:                      name:=schl.name;
[YR2] 1Ø87:                      time:=schl.time;
[EV2] 1Ø88:                      attr:=schl.attr;
[VS2] 1Ø89:                      size:=schl.size
[4MØ] 1Ø9Ø:                      end
[4MØ] 1Ø91:                 end
[ZNØ] 1Ø92:              end;
[KD2] 1Ø93:       findnext(schl)
[ZNØ] 1Ø94:       end;
[DM5] 1Ø95:    if doserror<>18 then infeld:=false
[DZ2] 1Ø96:                    else infeld:=true
[ZNØ] 1Ø97:    end;
[ØØØ] 1Ø98:
[MUØ] 1Ø99:  begin
[XD2] 11ØØ:  isopen:=false;
[253] 11Ø1:  if infeld(path) then
[MUØ] 11Ø2:     begin
[8N3] 11Ø3:     v:=scanright(path,'\');
[NB1] 11Ø4:     u:=path;
[MT3] 11Ø5:     delright(u,length(v)+1);
[YS4] 11Ø6:     if dfsize>Ø then least:=true;
[VPØ] 11Ø7:     i:=1;
[LU5] 11Ø8:     while (i<=dfsize)and not abbruch do
[MUØ] 11Ø9:           begin
[WS4] 111Ø:           uebertrage(dateifeld[i],u);
[L12] 1111:           escpressed;
```

Listing 1. »sbackup.pas« ist das Turbo-Pascal-5.5-Hauptprogramm des Multi-Backup-Systems »SBackup«

```
[XXØ] 1112:            inc(i)
[ZNØ] 1113:            end;
[H54] 1114:      if abbruch then goto abbr;
[VPØ] 1115:      i:=1;
[ØØØ] 1116:
[F12] 1117:      if sflag then
[ØR5] 1118:        while (i<=dirsize)and not abbruch do
[MUØ] 1119:              begin
[EX1] 112Ø:              inc(dirno);
[8Y4] 1121:              backupdir(u+'\'+dirfeld[i]+'\'
[SL1] 1122:                        +v,error);
[XXØ] 1123:              inc(i)
[4MØ] 1124:              end
[ZNØ] 1125:      end;
[PSØ] 1126:    abbr:
[ZNØ] 1127:    end;
[ØØØ] 1128:
[6Y6] 1129: procedure nextdisk;  {nur für restore}
[ØF5] 113Ø: var ch:char; error,er,fd:boolean;
[S95] 1131:     a:byte; w:string[6]; label err;
[MUØ] 1132:  begin
[KØ2] 1133:  inc(diskno);
[TH3] 1134:  if indisk then smooth;
[Y24] 1135:  a:=textattr; on:=false;
[1R5] 1136:  textattr:=col2; normrahmen:=col2;
[YW2] 1137:  rahmen(2Ø,17,37,1,'');
[DC1] 1138:  nomaus;
[2W3] 1139:  clrscr; ersatz:=false;
[CN5] 114Ø:  write(' Bitte Diskette Nr. ',diskno,
[NW1] 1141:           ' einlegen');
[RD5] 1142:  infoline(' Taste oder Mausknopf'+
[UY3] 1143:           ' drücken, wenn fertig');
[BM1] 1144:  showmaus;
[RG4] 1145:  repeat until keypressed or
[CA3] 1146:         button1 or button2;
[GH2] 1147:  restorescreen;
[E65] 1148:  if keypressed then ch:=readkey;
[WS4] 1149:  if (ord(ch)=27) or button2 then
[MUØ] 115Ø:     begin
[XD2] 1151:     abbruch:=true;
[3E1] 1152:     goto err
[ZNØ] 1153:     end;
[WX2] 1154:  nobutton; clrbuf;
[P31] 1155:  repeat
[B31] 1156:  ch:='a';
[8R2] 1157:  readboot(error);
[NY3] 1158:  w:=right(bootsec.vol,6);
[733] 1159:  w:=scanright(w,'k');
[LH5] 116Ø:  if error then wiederholen(error);
[A86] 1161:  fd:=(wordval(asstring(w),er)<>diskno)
[U15] 1162:   and not abbruch and not error
[4U2] 1163:   and not ersatz;
[FT4] 1164:  if fd then abbruch:=falsedisk;
[LF5] 1165:  until not(error or fd) or abbruch;
[LY1] 1166:  infoline('');
[E14] 1167:  if abbruch then goto err;
[AH1] 1168:  lwreset;
[2G4] 1169:  reads(rlw,1,1,puffer,error);
[W63] 117Ø:  if error then begin
[ZB1] 1171:                errorh;
[3E1] 1172:                goto err
[ZNØ] 1173:                end;
[XH5] 1174:  freesektor:=bootsec.disksize-2;
[2F2] 1175:  nextsektor:=2;
[2SØ] 1176:  err:
[UK1] 1177:  on:=true;
[1L4] 1178:  textattr:=a; normrahmen:=a
[ZNØ] 1179:  end;
[ØØØ] 118Ø:
[867] 1181: procedure nextdisk1;  {nur für restore}
[SS6] 1182: var ch:char; error,er,fd:boolean; a:byte;
[414] 1183:     w:string[6]; label err;
[MUØ] 1184:  begin
[KØ2] 1185:  inc(diskno);
[TH3] 1186:  if indisk then smooth;
[Y24] 1187:  a:=textattr; on:=false;
[1R5] 1188:  textattr:=col2; normrahmen:=col2;
[YW2] 1189:  rahmen(2Ø,17,37,1,'');
[DC1] 119Ø:  nomaus;
[2W3] 1191:  clrscr; ersatz:=false;
[CN5] 1192:  write(' Bitte Diskette Nr. ',diskno,
[NW1] 1193:           ' einlegen');
[RD5] 1194:  infoline(' Taste oder Mausknopf'+
[UY3] 1195:           ' drücken, wenn fertig');
[BM1] 1196:  showmaus;
[RG4] 1197:  repeat until keypressed or
[CA3] 1198:         button1 or button2;
[GH2] 1199:  restorescreen;
[E65] 12ØØ:  if keypressed then ch:=readkey;
[WS4] 12Ø1:  if (ord(ch)=27) or button2 then
[MUØ] 12Ø2:     begin
[XD2] 12Ø3:     abbruch:=true;
[3E1] 12Ø4:     goto err
[ZNØ] 12Ø5:     end;
[WX2] 12Ø6:  nobutton; clrbuf;
[P31] 12Ø7:  repeat
[B31] 12Ø8:  ch:='a';
[8R2] 12Ø9:  readboot(error);
[NY3] 121Ø:  w:=right(bootsec.vol,6);
[733] 1211:  w:=scanright(w,'k');
```

```
[LH5] 1212:  if error then wiederholen(error);
[4U6] 1213:  fd:=(wordval(asstring(w),er)<>diskno) and
[YC4] 1214:   not abbruch and not error
[4U2] 1215:   and not ersatz;
[FT4] 1216:  if fd then abbruch:=falsedisk;
[LF5] 1217:  until not(error or fd) or abbruch;
[LY1] 1218:  infoline('');
[E14] 1219:  if abbruch then goto err;
[AH1] 122Ø:  lwreset;
[UH5] 1221:  freesektor:=bootsec.disksize-1;
[ØF2] 1222:  nextsektor:=1;
[2SØ] 1223:  err:
[UK1] 1224:  on:=true;
[1L4] 1225:  textattr:=a; normrahmen:=a
[ZNØ] 1226:  end;
[ØØØ] 1227:
[LH3] 1228: procedure nextsekt;
[FB6] 1229: var error:boolean; p:pointer; label 1;
[MUØ] 123Ø:  begin
[ØP1] 1231:  p:=puffer;
[T26] 1232:  inc(ptrec(p).ofs,bootsec.sektsize);
[PA5] 1233:  if nextsektor=bootsec.disksize
[SL2] 1234:     then nextdisk1;
[EP3] 1235:  if abbruch then goto 1;
[SC5] 1236:  reads(rlw,nextsektor,1,p,error);
[BF3] 1237:  if error then errorh;
[MG5] 1238:  inc(freebyte,bootsec.sektsize);
[RT2] 1239:  inc(nextsektor);
[EP2] 124Ø:  dec(freesektor);
[G8Ø] 1241:  1:
[ZNØ] 1242:  end;
[ØØØ] 1243:
[6K3] 1244: procedure nextsekt1;
[UF4] 1245: var error:boolean;  label 1;
[MUØ] 1246:  begin
[PA5] 1247:  if nextsektor=bootsec.disksize
[SL2] 1248:     then nextdisk1;
[EP3] 1249:  if abbruch then goto 1;
[3A6] 125Ø:  reads(rlw,nextsektor,1,puffer,error);
[BF3] 1251:  if error then errorh;
[3Z4] 1252:  freebyte:=bootsec.sektsize;
[RT2] 1253:  inc(nextsektor);
[EP2] 1254:  dec(freesektor);
[G8Ø] 1255:  1:
[ZNØ] 1256:  end;
[ØØØ] 1257:
[KV3] 1258: procedure ruebertrage;
[AF5] 1259: type prec=record size: longint;
[SX1] 126Ø:                  attr: byte;
[U62] 1261:                  time: longint
[ZNØ] 1262:                  end;
[SK5] 1263: var s,path:string; size:longint;
[DC4] 1264:     l:byte; p1: ^byte; r1:prec;
[YZ3] 1265:     a,b,c,d:word; p:pointer;
[RN3] 1266:     reg:registers; f:file;
[HF4] 1267:     er:boolean; label err1,err2;
[ØØØ] 1268:
[MUØ] 1269:  begin
[G82] 127Ø:  error:=false;
[CT4] 1271:  if freebyte<5 then nextsekt1;
[D64] 1272:  if abbruch then goto err2;
[A84] 1273:  inc(ptrec(sektpoi).ofs,5);
[ØL2] 1274:  dec(freebyte,5);
[T12] 1275:  p1:=sektpoi;
[K71] 1276:  l:=p1^;
[ZX4] 1277:  if freebyte<l+1 then nextsekt;
[D64] 1278:  if abbruch then goto err2;
[XG3] 1279:  move(sektpoi^,s,l+1);
[6X2] 128Ø:  dec(freebyte,l+1);
[6F4] 1281:  inc(ptrec(sektpoi).ofs,l+1);
[BT4] 1282:  if freebyte<=9 then nextsekt;
[D64] 1283:  if abbruch then goto err2;
[CL2] 1284:  dec(freebyte,9);
[4C3] 1285:  move(sektpoi^,r1,9);
[E84] 1286:  inc(ptrec(sektpoi).ofs,9);
[GT6] 1287:  if ptrec(sektpoi).ofs>=bootsec.sektsize
[ZN1] 1288:     then begin
[ØP1] 1289:          p:=puffer;
[8H5] 129Ø:          ptrec(p).ofs:=bootsec.sektsize;
[913] 1291:          move(p^,puffer^,
[P63] 1292:               bootsec.sektsize);
[DX3] 1293:          dec(ptrec(sektpoi).ofs,
[513] 1294:              bootsec.sektsize)
[ZNØ] 1295:          end;
[ØØØ] 1296:
[DD1] 1297:  path:=s;
[2N3] 1298:  s:=scanright(path,'\');
[ZD4] 1299:  delright(path,length(s)+1);
[ØF2] 13ØØ:  size:=r1.size;
[MØ4] 13Ø1:  if aktpath<>pseudo(path)
[194] 13Ø2:     then changedir(path,fehl);
[VW2] 13Ø3:  if fehl then begin
[2K2] 13Ø4:               verzeicherror;
[7H1] 13Ø5:               goto err2
[ZNØ] 13Ø6:               end;
[DC1] 13Ø7:  nomaus;
[DB1] 13Ø8:  clrscr;
[X95] 13Ø9:  if right(pseudo(path),1)='\' then
[T26] 131Ø:   write(upstring(pseudo(path))+s) else
[RY5] 1311:   write(upstring(pseudo(path))+'\'+s);
```

```
[BM1] 1312:  showmaus;
[YP3] 1313:  zdatei:=make(s,Ø,fehl);
[YX1] 1314:  if fehl then
[MUØ] 1315:     begin
[ZB1] 1316:     errorh;
[FØ5] 1317:     while (size>Ø)and not abbruch do
[MUØ] 1318:           begin
[3L3] 1319:           if size<freebyte then
[MUØ] 132Ø:              begin
[AA3] 1321:              dec(freebyte,size);
[T81] 1322:              size:=Ø
[4MØ] 1323:              end
[ZRØ] 1324:           else
[MUØ] 1325:              begin
[9B3] 1326:              dec(size,freebyte);
[F35] 1327:              a:=size div bootsec.sektsize;
[9G3] 1328:              if a>=freesektor then
[MUØ] 1329:                 begin
[TK3] 133Ø:                 dec(size,freesektor*
[P63] 1331:                     bootsec.sektsize);
[EC2] 1332:                 indisk:=false;
[HH3] 1333:                 nextdisk1; freebyte:=
[923] 1334:                     bootsec.sektsize;
[D64] 1335:                 if abbruch then goto err2;
[EP2] 1336:                 dec(freesektor);
[RT2] 1337:                 inc(nextsektor);
[4MØ] 1338:                 end
[ZRØ] 1339:              else
[MUØ] 134Ø:                 begin
[853] 1341:                 dec(freesektor,a+1);
[AT1] 1342:                 freebyte:=
[923] 1343:                      bootsec.sektsize;
[H83] 1344:                 inc(nextsektor,a+1);
[RA3] 1345:                 dec(size,longmul(a,
[CC3] 1346:                     bootsec.sektsize));
[4MØ] 1347:                 end
[4MØ] 1348:              end;
[ZNØ] 1349:           end;
[D64] 135Ø:     if abbruch then goto err2;
[SF5] 1351:     reads(rlw,nextsektor-1,1,puffer,
[291] 1352:           error);
[2Y2] 1353:     sektpoi:=puffer;
[AB6] 1354:     ptrec(sektpoi).ofs:=bootsec.sektsize
[WR1] 1355:                        -freebyte;
[7H1] 1356:     goto err2
[ZNØ] 1357:     end;
[ØØØ] 1358:
[8L2] 1359:  while size>Ø do
[MUØ] 136Ø:    begin
[U35] 1361:    nochfrei:=freebyte+freesektor*
[923] 1362:              bootsec.sektsize;
[WA4] 1363:    if size<$faØØ+freebyte then
[MUØ] 1364:       begin
[VN3] 1365:       if nochfrei>=size then
[MUØ] 1366:          begin
[5L3] 1367:          if size>freebyte then
[MUØ] 1368:             begin
[9B3] 1369:             dec(size,freebyte);
[F35] 137Ø:             a:=size div bootsec.sektsize;
[D25] 1371:             b:=size mod bootsec.sektsize;
[ØP1] 1372:             p:=puffer;
[CU2] 1373:             inc(ptrec(p).ofs,
[P63] 1374:                 bootsec.sektsize);
[MY3] 1375:             if b=Ø then d:=Ø else d:=1;
[WL4] 1376:             reads(rlw,nextsektor,a+d,p,
[291] 1377:                   error);
[CU3] 1378:             if error then goto err1;
[AA3] 1379:             inc(nextsektor,a+d);
[173] 138Ø:             dec(freesektor,a+d);
[R53] 1381:             inc(ptrec(p).ofs,a*
[P63] 1382:                 bootsec.sektsize);
[WR3] 1383:             c:=writeh(zdatei,size+
[W54] 1384:                   freebyte,sektpoi^,fehl);
[AR3] 1385:             if fehl then goto err1;
[634] 1386:             if b=Ø then nextsekt1 else
[MUØ] 1387:                begin
[913] 1388:                move(p^,puffer^,
[P63] 1389:                     bootsec.sektsize);
[RR4] 139Ø:                freebyte:=bootsec.sektsize
[Ø9Ø] 1391:                          -b
[ZNØ] 1392:                end;
[2Y2] 1393:             sektpoi:=puffer;
[R84] 1394:             inc(ptrec(sektpoi).ofs,b);
[GZ4] 1395:             if freesektor<Ø then nextdisk;
[B64] 1396:             if abbruch then goto err1;
[T81] 1397:             size:=Ø
[4MØ] 1398:             end
[ZRØ] 1399:          else
[V44] 14ØØ:             begin  {size<=freebyte}
[ZR3] 14Ø1:             c:=writeh(zdatei,size,
[9M2] 14Ø2:                       sektpoi^,fehl);
[AR3] 14Ø3:             if fehl then goto err1;
[AA3] 14Ø4:             dec(freebyte,size);
[SV4] 14Ø5:             inc(ptrec(sektpoi).ofs,size);
[T81] 14Ø6:             size:=Ø
[4MØ] 14Ø7:             end
[4MØ] 14Ø8:          end
[ZRØ] 14Ø9:       else
[M74] 141Ø:          begin   {nochfrei<size}
[ØP1] 1411:          p:=puffer;
[CU2] 1412:          inc(ptrec(p).ofs,
```

```
[P63] 1413:                bootsec.sektsize);
[E93] 1414:          if freesektor>Ø then
[VS5] 1415:          reads(rlw,nextsektor,freesektor,
[2G1] 1416:                p,error);
[CU3] 1417:          if error then goto err1;
[C14] 1418:          a:=freebyte+freesektor*
[923] 1419:             bootsec.sektsize;
[8V4] 142Ø:          c:=writeh(zdatei,a,sektpoi^,
[AYØ] 1421:                    fehl);
[AR3] 1422:          if fehl then goto err1;
[CT1] 1423:          nextdisk1;
[B64] 1424:          if abbruch then goto err1;
[LX1] 1425:          dec(size,a);
[F35] 1426:          a:=size div bootsec.sektsize;
[ZS4] 1427:          reads(rlw,1,a+1,puffer,error);
[CU3] 1428:          if error then goto err1;
[S45] 1429:          c:=writeh(zdatei,size,puffer^,
[AYØ] 143Ø:                    fehl);
[AR3] 1431:          if fehl then goto err1;
[ØP1] 1432:          p:=puffer;
[R53] 1433:          inc(ptrec(p).ofs,a*
[P63] 1434:              bootsec.sektsize);
[913] 1435:          move(p^,puffer^,
[P63] 1436:               bootsec.sektsize);
[2Y2] 1437:          sektpoi:=puffer;
[D25] 1438:          b:=size mod bootsec.sektsize;
[R84] 1439:          inc(ptrec(sektpoi).ofs,b);
[335] 144Ø:          freebyte:=bootsec.sektsize-b;
[853] 1441:          dec(freesektor,a+1);
[H83] 1442:          inc(nextsektor,a+1);
[T81] 1443:          size:=Ø
[4MØ] 1444:          end
[4MØ] 1445:       end
[ZRØ] 1446:    else
[WP4] 1447:       begin  {size>=$faØØ+freebyte}
[VN3] 1448:       if nochfrei>=size then
[MUØ] 1449:          begin
[4X4] 145Ø:          a:=$faØØ div bootsec.sektsize;
[ØP1] 1451:          p:=puffer;
[CU2] 1452:          inc(ptrec(p).ofs,
[P63] 1453:              bootsec.sektsize);
[8E5] 1454:          reads(rlw,nextsektor,a,p,error);
[CU3] 1455:          if error then goto err1;
[S65] 1456:          c:=writeh(zdatei,$faØØ+freebyte,
[9M2] 1457:                    sektpoi^,fehl);
[AR3] 1458:          if fehl then goto err1;
[853] 1459:          dec(freesektor,a+1);
[H83] 146Ø:          inc(nextsektor,a+1);
[2Z3] 1461:          dec(size,$faØØ+freebyte);
[ZT4] 1462:          if freesektor<Ø then nextdisk
[SY4] 1463:             else reads(rlw,nextsektor-1,1,
[XC2] 1464:                        puffer,error);
[V83] 1465:          if error or abbruch
[EG2] 1466:             then goto err1;
[2Y2] 1467:          sektpoi:=puffer;
[RR4] 1468:          freebyte:=bootsec.sektsize
[4MØ] 1469:          end
[ZRØ] 147Ø:       else
[VU3] 1471:          begin  {nochfrei<size}
[ØP1] 1472:          p:=puffer;
[CU2] 1473:          inc(ptrec(p).ofs,
[P63] 1474:              bootsec.sektsize);
[4X4] 1475:          a:=$faØØ div bootsec.sektsize;
[1E3] 1476:          if a>freesektor then
[MUØ] 1477:             begin
[E93] 1478:             if freesektor>Ø then
[HP3] 1479:               reads(rlw,nextsektor,
[SC3] 148Ø:                     freesektor,p,error);
[CU3] 1481:             if error then goto err1;
[KB2] 1482:             b:=freesektor*
[923] 1483:                bootsec.sektsize;
[ØT4] 1484:             c:=writeh(zdatei,freebyte+b,
[9M2] 1485:                       sektpoi^,fehl);
[AR3] 1486:             if fehl then goto err1;
[LF3] 1487:             dec(size,b+freebyte);
[FM1] 1488:             nextdisk;
[B64] 1489:             if abbruch then goto err1;
[2Y2] 149Ø:             sektpoi:=puffer;
[3Z4] 1491:             freebyte:=bootsec.sektsize;
[4MØ] 1492:             end
[ZRØ] 1493:          else
[K44] 1494:             begin   {a<=freesektor}
[C74] 1495:             reads(rlw,nextsektor,a,p,
[291] 1496:                   error);
[CU3] 1497:             if error then goto err1;
[CF4] 1498:             c:=writeh(zdatei,freebyte+
[RD3] 1499:                       $faØØ,sektpoi^,fehl);
[AR3] 15ØØ:             if fehl then goto err1;
[853] 15Ø1:             dec(freesektor,a+1);
[H83] 15Ø2:             inc(nextsektor,a+1);
[2Z3] 15Ø3:             dec(size,$faØØ+freebyte);
[ZT4] 15Ø4:             if freesektor<Ø then nextdisk
[KR4] 15Ø5:               else reads(rlw,nextsektor-1,
[LL2] 15Ø6:                          1,puffer,error);
[W33] 15Ø7:             if abbruch or error
[EG2] 15Ø8:                then goto err1;
[2Y2] 15Ø9:             sektpoi:=puffer;
[RR4] 151Ø:             freebyte:=bootsec.sektsize
```

Listing 1. »sbackup.pas« ist das Turbo-Pascal-5.5-Hauptprogramm des Multi-Backup-Systems »SBackup«

```
[4MØ] 1511:              end
[4MØ] 1512:            end
[4MØ] 1513:          end
[ZNØ] 1514:      end;
[ØØØ] 1515:
[UT1] 1516:  reg.ah:=$57;
[VK1] 1517:  reg.al:=1;
[RM2] 1518:  reg.bx:=zdatei;
[BP4] 1519:  reg.cx:=longrec(r1.time).lo;
[9N4] 152Ø:  reg.dx:=longrec(r1.time).hi;
[KW1] 1521:  msdos(reg);
[DVØ] 1522:  err1:
[MX2] 1523:  closeh(zdatei,er);
[SH4] 1524:  if error or fehl then begin
[ZB1] 1525:                          errorh;
[LW2] 1526:                          deleteh(s,error);
[7H1] 1527:                          goto err2
[ZNØ] 1528:                          end;
[YW1] 1529:  assign(f,s);
[UL2] 153Ø:  {$i-} reset(f);
[4U1] 1531:  a:=r1.attr;
[1A2] 1532:  setfattr(f,a);
[9T2] 1533:  close(f); {$i+}
[EVØ] 1534:  err2:
[ZNØ] 1535:  end;
[ØØØ] 1536:
[L73] 1537: procedure restore;
[UE6] 1538: var ende: boolean; pos:word; label abbr;
[MUØ] 1539:  begin
[LL1] 154Ø:  elpstime;
[D12] 1541:  ende:=false;
[CL2] 1542:  abbruch:=false;
[TN1] 1543:  diskno:=Ø;
[V62] 1544:  restmeldung;
[EC2] 1545:  indisk:=false;
[CT1] 1546:  nextdisk1;
[E92] 1547:  indisk:=true;
[H54] 1548:  if abbruch then goto abbr;
[2G4] 1549:  reads(rlw,1,1,puffer,error);
[W63] 155Ø:  if error then begin
[ZB1] 1551:                 errorh;
[BG1] 1552:                 goto abbr
[ZNØ] 1553:                 end;
[2Y2] 1554:  sektpoi:=puffer;
[3Z4] 1555:  freebyte:=bootsec.sektsize;
[RT2] 1556:  inc(nextsektor);
[EP2] 1557:  dec(freesektor);
[3P4] 1558:  while not (ende or abbruch) do
[MUØ] 1559:        begin
[CN3] 156Ø:        if ptrec(sektpoi).ofs>
[R14] 1561:           bootsec.sektsize-5 then
[MUØ] 1562:           begin
[7T1] 1563:           nextsekt1;
[2Y2] 1564:           sektpoi:=puffer;
[ØØ4] 1565:           if abbruch then goto abbr
[ZNØ] 1566:           end;
[AG5] 1567:        abbruch:=not searchsektor(ptrec
[YW3] 1568:                  (sektpoi).ofs,pos,ende);
[4G4] 1569:        if not (abbruch or ende) then
[CR2] 157Ø:           begin  {if found}
[B54] 1571:           ptrec(sektpoi).ofs:=pos;
[162] 1572:           ruebertrage;
[7U1] 1573:           escpressed
[4MØ] 1574:           end
[ZNØ] 1575:        end;
[UN1] 1576:  on:=false;
[LH5] 1577:  if not abbruch or ende then fertig;
[7YØ] 1578:  abbr:;
[G74] 1579:  restorescreen; on:=false;
[XP5] 158Ø:  textattr:=col1; normrahmen:=col1;
[HE4] 1581:  restorescreen  {elpstime}
[ZNØ] 1582:  end;
[ØØØ] 1583:
[884] 1584: procedure backuphandler;
[KD6] 1585: var j,i,size,ch:byte; err,min:boolean;
[WA1] 1586:     label ab;
[MUØ] 1587:  begin
[4Z1] 1588:  min:=false;
[MH2] 1589:  for i:=1 to 2Ø do
[WF5] 159Ø:      if lwfeld[i]<>nil then min:=true;
[Ø93] 1591:  if not min then exit;
[LL1] 1592:  elpstime;
[VN1] 1593:  diskno:=1;
[KT2] 1594:  restflag:=false;
[Z53] 1595:  setverify(verify);
[BM4] 1596:  abbruch:=false; err:=false;
[632] 1597:  backmeldung;
[YF1] 1598:  insdisk;
[NU3] 1599:  if abbruch then goto ab;
[H14] 16ØØ:  on:=true; least:=false;
[MH2] 16Ø1:  for i:=1 to 2Ø do
[4G5] 16Ø2:     if (lwfeld[i]<>nil)and not abbruch
[ZU2] 16Ø3:        and not err then
[MUØ] 16Ø4:        begin
[Ø14] 16Ø5:        size:=lwfeld[i]^.size;
[XPØ] 16Ø6:        j:=1;
[514] 16Ø7:        while (j<=size)and not err
[Z23] 16Ø8:              and not abbruch do
[MUØ] 16Ø9:              begin
[ØX4] 161Ø:              sflag:=lwfeld[i]^.sflag[j];
[545] 1611:              backupdir(lwfeld[i]^.sub[j],
```

```
[7VØ] 1612:                              err);
[ZXØ] 1613:              inc(j)
[4MØ] 1614:              end
[ZNØ] 1615:        end;
[NU3] 1616:  if abbruch then goto ab;
[724] 1617:  move(endrec,sektpoi^,6);
[3L2] 1618:  dec(freebyte,6);
[L64] 1619:  if freebyte<Ø then nexts1;
[HF4] 162Ø:  if not abbruch and least then
[MUØ] 1621:     begin
[1X5] 1622:     writes(lw,nextsektor,zuschreiben,
[XC2] 1623:            puffer,error);
[BF3] 1624:     if error then errorh;
[US3] 1625:     if abbruch then goto ab
[ZNØ] 1626:     end;
[UN1] 1627:  on:=false;
[2P2] 1628:  if not least then
[MUØ] 1629:     begin
[5R5] 163Ø:     textattr:=col3; normrahmen:=col3;
[AK4] 1631:     rahmen(25,11,3Ø,3,''); on:=false;
[DC1] 1632:     nomaus;
[UT3] 1633:     clrscr; writeln; beep;
[ØF5] 1634:     write('  Keine Dateien gefunden !');
[YE5] 1635:     infoline(' Taste oder Maustaste'+
[EV1] 1636:              ' drücken');
[BM1] 1637:     showmaus;
[RG4] 1638:     repeat until keypressed or
[CA3] 1639:            button1 or button2;
[9A4] 164Ø:     if keypressed then clrbuf;
[LY1] 1641:     infoline('');
[GH2] 1642:     restorescreen;
[MG1] 1643:     nobutton
[NN3] 1644:     end             else fertig;
[ØØØ] 1645:
[GHØ] 1646:  ab:
[UN1] 1647:  on:=false;
[GH2] 1648:  restorescreen;
[XP5] 1649:  textattr:=col1; normrahmen:=col1;
[2W4] 165Ø:  restorescreen     {elpstime}
[ZNØ] 1651:  end;
[ØØØ] 1652:
[XX6] 1653: procedure filerestore(teilflag:boolean);
[VE1] 1654:           forward;
[ØØØ] 1655:
[YZ3] 1656: procedure teilrestore;
[2K5] 1657: var ende,found,error,er:boolean;
[FM4] 1658:     i,anf,ps,pos:word; label 1,ab;
[MUØ] 1659:  begin
[T73] 166Ø:  filerestore(true);
[EP3] 1661:  if abbruch then goto 1;
[V62] 1662:  restmeldung;
[965] 1663:  i:=Ø; found:=false; ende:=false;
[CA5] 1664:  while (i<dfsize)and not found do
[MUØ] 1665:    begin
[951] 1666:    inc(i);
[873] 1667:    found:=selected[i]
[ZNØ] 1668:    end;
[UK1] 1669:  on:=true;
[Y14] 167Ø:  if not found then goto ab;
[H96] 1671:  reads(rlw,sektfeld[i].sekt,1,puffer,
[291] 1672:        error);
[W63] 1673:  if error then begin
[ZB1] 1674:                 errorh;
[591] 1675:                 goto ab
[ZNØ] 1676:                 end;
[2Y2] 1677:  sektpoi:=puffer;
[LS6] 1678:  inc(ptrec(sektpoi).ofs,sektfeld[i].ofs);
[BX4] 1679:  freebyte:=bootsec.sektsize-
[EZ2] 168Ø:             sektfeld[i].ofs;
[EM5] 1681:  nextsektor:=sektfeld[i].sekt+1;
[997] 1682:  freesektor:=bootsec.disksize-nextsektor;
[ØØØ] 1683:
[3P4] 1684:  while not (ende or abbruch) do
[MUØ] 1685:    begin
[CN3] 1686:    if ptrec(sektpoi).ofs>
[R14] 1687:       bootsec.sektsize-5 then
[MUØ] 1688:       begin
[7T1] 1689:       nextsekt1;
[2Y2] 169Ø:       sektpoi:=puffer;
[US3] 1691:       if abbruch then goto ab
[ZNØ] 1692:       end;
[3D5] 1693:    abbruch:= not searchsektor(ptrec
[YW3] 1694:                 (sektpoi).ofs,pos,ende);
[4G4] 1695:    if not (abbruch or ende) then
[CR2] 1696:       begin  {if found}
[B54] 1697:       ptrec(sektpoi).ofs:=pos;
[162] 1698:       ruebertrage;
[7U1] 1699:       escpressed
[4MØ] 17ØØ:       end
[ZNØ] 17Ø1:   end;
[UN1] 17Ø2:  on:=false;
[7D4] 17Ø3:  if not abbruch then fertig;
[GHØ] 17Ø4:  ab:
[GH2] 17Ø5:  restorescreen;
[G35] 17Ø6:  restorescreen;   {elpstime}
[G8Ø] 17Ø7:  1:
[XP5] 17Ø8:  textattr:=col1; normrahmen:=col1;
[9G1] 17Ø9:  on:=false
[ZNØ] 171Ø:  end;
[ØØØ] 1711:
```

```
[XX6] 1712: procedure filerestore(teilflag:boolean);
[H66] 1713: var k,i,attr,curspos,ch,ypos,orand,
[GG5] 1714:     urand:byte; fert,onefile,ende,er,
[824] 1715:     error: boolean; label 1,2;
[ØØØ] 1716:
[TZ2] 1717:      procedure invers;
[MUØ] 1718:      begin
[EV2] 1719:      attr:=textattr;
[4L4] 172Ø:      textattr:=selectattribut;
[6N2] 1721:      gotoxy(2,ypos);
[N84] 1722:      write(subdrs[curspos]);
[CL2] 1723:      textattr:=attr
[ZNØ] 1724:      end;
[ØØØ] 1725:
[5H3] 1726:      procedure writeups;
[BY1] 1727:      var a:byte;
[MUØ] 1728:      begin
[262] 1729:      a:=textattr;
[RR2] 173Ø:      textattr:=col6;
[LF4] 1731:      gotoxy(3,16); write(chr(24));
[UF4] 1732:      gotoxy(7,16); write(chr(25));
[GV1] 1733:      gotoxy(1,1);
[U32] 1734:      textattr:=a
[ZNØ] 1735:      end;
[ØØØ] 1736:
[MUØ] 1737: begin
[LL1] 1738: elpstime;
[CL2] 1739: abbruch:=false;
[AL1] 174Ø: insdisk1;
[E92] 1741: indisk:=true;
[EP3] 1742: if abbruch then goto 1;
[2F4] 1743: diskno:=wordval(scanright
[YE3] 1744:         (bootsec.vol,'k'),er);
[DC1] 1745: nomaus;
[6U5] 1746: textattr:=col4; normrahmen:= col4;
[3X2] 1747: rahmen(25,11,27,3,'');
[DB1] 1748: clrscr;
[XF1] 1749: writeln;
[KU5] 175Ø: writeln('  Lese Disketteninfos...');
[BM1] 1751: showmaus;
[7K2] 1752: scandisk(ende);
[GH2] 1753: restorescreen;
[EP3] 1754: if abbruch then goto 1;
[4Z3] 1755: if ende then dec(dfsize);
[EH2] 1756: if dfsize=Ø then
[MUØ] 1757:    begin
[5R5] 1758:    textattr:=col3; normrahmen:=col3;
[3K4] 1759:    rahmen(2Ø,11,4Ø,3,''); on:=false;
[DC1] 176Ø:    nomaus;
[UT3] 1761:    clrscr; writeln; beep;
[YN5] 1762:    write(' Keine verfügbaren Files'+
[ØL2] 1763:          ' auf Diskette!');
[YE5] 1764:    infoline(' Taste oder Maustaste'+
[EV1] 1765:             ' drücken');
[BM1] 1766:    showmaus;
[RG4] 1767:    repeat until keypressed or
[CA3] 1768:           button1 or button2;
[9A4] 1769:    if keypressed then clrbuf;
[LY1] 177Ø:    infoline('');
[GH2] 1771:    restorescreen;
[AM1] 1772:    nobutton;
[W23] 1773:    abbruch:=teilflag;
[DZØ] 1774:    goto 1
[ZNØ] 1775:    end;
[DC1] 1776: nomaus;
[KD3] 1777: for i:=1 to dfsize do
[E85] 1778:     subdrs[i]:=dateifeld[i].name;
[XP5] 1779: textattr:=col1; normrahmen:=col1;
[4W6] 178Ø: for i:=1 to dfsize do selected[i]:=false;
[NW5] 1781: rahmen(25,5,3Ø,16,'Files auf Diskette');
[DB1] 1782: clrscr;
[L35] 1783: fert:=false;  ch:=Ø; curspos:=1;
[5F5] 1784: ypos:=1; orand:=1; onefile:=false;
[4V5] 1785: urand:=15; writefiles(orand,urand);
[EN1] 1786: writeups;
[2D1] 1787: invers;
[R56] 1788: infoline(' Bitte Dateien auswählen,'+
[KZ3] 1789:          ' <ESC> oder linke Maust'+
[V73] 179Ø:          'aste, wenn fertig');
[BM1] 1791: showmaus;
[EP2] 1792: while not fert do
[MUØ] 1793:   begin
[ØT5] 1794:   if keypressed then ch:=ord(readkey);
[YG3] 1795:   if (ch=72)or(ch=8Ø) then
[MUØ] 1796:      begin
[6N2] 1797:      gotoxy(2,ypos);
[DC1] 1798:      nomaus;
[N84] 1799:      write(subdrs[curspos]);
[VX1] 18ØØ:      if ch=8Ø then
[MUØ] 181Ø:         begin
[VX2] 182Ø:         if curspos<dfsize
[Z23] 183Ø:            then inc(curspos);
[7L3] 184Ø:         if curspos>urand then
[MUØ] 185Ø:            begin
[DB1] 186Ø:            clrscr;
[5W1] 187Ø:            inc(orand);
[HW1] 188Ø:            inc(urand);
[914] 189Ø:            writefiles(orand,urand)
[4MØ] 191Ø:            end
[ZNØ] 1911:         end;
```

```
[TX1] 1812:       if ch=72 then
[MUØ] 1813:          begin
[D55] 1814:          if curspos>1 then dec(curspos);
[TK3] 1815:          if curspos<orand then
[MUØ] 1816:             begin
[DB1] 1817:             clrscr;
[ØV1] 1818:             dec(orand);
[CV1] 1819:             dec(urand);
[914] 182Ø:             writefiles(orand,urand)
[4MØ] 1821:             end
[ZNØ] 1822:          end;
[YU3] 1823:       ypos:=curspos-orand+1;
[2D1] 1824:       invers;
[EN1] 1825:       writeups;
[BM1] 1826:       showmaus;
[3RØ] 1827:       ch:=Ø
[ZNØ] 1828:       end;
[HZ3] 1829:    if (ch=27)or button2 then
[MUØ] 183Ø:       begin
[912] 1831:       fert:=true;
[U54] 1832:       repeat until not button2
[ZNØ] 1833:       end;
[5G2] 1834:    if button1 then
[D92] 1835:     if akty>4 then
[ØE2] 1836:      if aktx>24 then
[PD2] 1837:       if aktx<4Ø then
[PD2] 1838:        if akty<2Ø then
[MUØ] 1839:           begin
[H34] 184Ø:           if curspos+akty-4-ypos<=
[XY1] 1841:              dfsize then
[MUØ] 1842:              begin
[DC1] 1843:              nomaus;
[6N2] 1844:              gotoxy(2,ypos);
[N84] 1845:              write(subdrs[curspos]);
[R55] 1846:              curspos:=curspos+akty-4-ypos;
[W52] 1847:              ypos:=akty-4;
[A31] 1848:              ch:=13;
[BM1] 1849:              showmaus;
[T54] 185Ø:              repeat until not button1
[4MØ] 1851:              end
[4MØ] 1852:           end
[G93] 1853:         else if akty=2Ø then
[8E2] 1854:               if aktx=27 then
[MUØ] 1855:                  begin
[GC2] 1856:                  if orand>1 then
[MUØ] 1857:                     begin
[DC1] 1858:                     nomaus;
[DB1] 1859:                     clrscr;
[ØV1] 186Ø:                     dec(orand);
[CV1] 1861:                     dec(urand);
[392] 1862:                     dec(curspos);
[NW2] 1863:                     writefiles(orand,
[171] 1864:                              urand);
[G33] 1865:                     invers; writeups;
[BM1] 1866:                     showmaus;
[1E1] 1867:                     delay(1ØØ)
[4MØ] 1868:                     end
[4MØ] 1869:                  end
[ZRØ] 187Ø:               else
[SD2] 1871:                if aktx=31 then
[MUØ] 1872:                   begin
[1C3] 1873:                   if urand<dfsize then
[MUØ] 1874:                      begin
[DC1] 1875:                      nomaus;
[DB1] 1876:                      clrscr;
[5W1] 1877:                      inc(orand);
[HW1] 1878:                      inc(urand);
[8A2] 1879:                      inc(curspos);
[NW2] 188Ø:                      writefiles(orand,
[171] 1881:                                urand);
[G33] 1882:                      invers; writeups;
[BM1] 1883:                      showmaus;
[1E1] 1884:                      delay(1ØØ)
[4MØ] 1885:                      end
[ZNØ] 1886:                   end;
[2X1] 1887:    if ch=13 then
[M73] 1888:     if not teilflag then
[MUØ] 1889:        begin
[DC1] 189Ø:        nomaus;
[6N2] 1891:        gotoxy(2,ypos);
[N84] 1892:        write(subdrs[curspos]);
[5R5] 1893:        infoline(' Bitte Dateien auswähl'+
[TE5] 1894:         'en, Pfeilzeichen anklicken, um'+
[UV3] 1895:         ' mit Maus zu scrollen');
[AL3] 1896:        selected[curspos]:=
[K34] 1897:             not selected[curspos];
[HB4] 1898:        if selected[curspos] then
[2A6] 1899:           subdrs[curspos]:=subdrs[curspos]
[MMØ] 19ØØ:             +' √'
[ZRØ] 19Ø1:           else
[VY4] 19Ø2:             delright(subdrs[curspos],2);
[DB1] 19Ø3:        clrscr;
[434] 19Ø4:        writefiles(orand,urand);
[G33] 19Ø5:        invers; writeups;
[BM1] 19Ø6:        showmaus;
[3RØ] 19Ø7:        ch:=Ø
[ØC2] 19Ø8:        end else begin
[912] 19Ø9:                  fert:=true;
```

Listing 1. »sbackup.pas« ist das Turbo-Pascal-5.5-Hauptprogramm des Multi-Backup-Systems »SBackup«

```
[NB4] 191Ø:                 selected[curspos]:=true
[4MØ] 1911:                 end
[ZNØ] 1912:     end;
[GH2] 1913:  restorescreen;
[LY1] 1914:  infoline('');
[KT3] 1915:  if teilflag then goto 2;
[V62] 1916:  restmeldung;
[UK1] 1917:  on:=true;
[KD3] 1918:  for i:=1 to dfsize do
[9M5] 1919:   if selected[i] and not abbruch then
[MUØ] 192Ø:      begin
[SY4] 1921:      reads(rlw,sektfeld[i].sekt,1,
[XC2] 1922:            puffer,error);
[W63] 1923:      if error then begin
[ZB1] 1924:                    errorh;
[GH2] 1925:                    restorescreen;
[DZØ] 1926:                    goto 1
[ZNØ] 1927:                    end;
[2Y2] 1928:      sektpoi:=puffer;
[KY3] 1929:      inc(ptrec(sektpoi).ofs,
[K23] 193Ø:          sektfeld[i].ofs);
[BX4] 1931:      freebyte:=bootsec.sektsize-
[EZ2] 1932:                sektfeld[i].ofs;
[EM5] 1933:      nextsektor:=sektfeld[i].sekt+1;
[V55] 1934:      freesektor:=bootsec.disksize-
[M42] 1935:                 nextsektor;
[MF2] 1936:      onefile:=true;
[162] 1937:      ruebertrage;
[7U1] 1938:      escpressed
[ZNØ] 1939:      end;
[UN1] 194Ø:  on:=false;
[K66] 1941:  if onefile and not abbruch then fertig;
[GH2] 1942:  restorescreen;
[G8Ø] 1943:  1:
[9T4] 1944:  restorescreen;  {elpstime}
[K8Ø] 1945:  2:
[UN1] 1946:  on:=false;
[XP5] 1947:  textattr:=col1; normrahmen:=col1;
[ZNØ] 1948:  end;
[ØØØ] 1949:
[MD4] 195Ø: procedure restorehandler;
[WZ3] 1951: var a:menufeld; q:byte;
[MUØ] 1952:  begin
[ØS2] 1953:  restflag:=true;
[2Z2] 1954:  getdir(Ø,aktpath);
[Z53] 1955:  setverify(verify);
[Z22] 1956:  q:=menumode;
[SZ1] 1957:  menumode:=Ø;
[HT2] 1958:  a[1]:='^Restore';
[WG3] 1959:  a[2]:='^Teilrestore';
[NG3] 196Ø:  a[3]:='^Filerestore';
[6L3] 1961:  verticalmenu(15,3,3,a);
[613] 1962:  if not escflag then
[ØR2] 1963:     case menumode of
[3V1] 1964:     1: restore;
[EK2] 1965:     2: teilrestore;
[BH3] 1966:     3: filerestore(false);
[ZNØ] 1967:      end;
[GH2] 1968:  restorescreen;
[EC2] 1969:  indisk:=false;
[122] 197Ø:  menumode:=q
[ZNØ] 1971:  end;
[ØØØ] 1972:
[2W2] 1973: procedure intro;
[BY1] 1974: var a:byte;
[MUØ] 1975:  begin
[262] 1976:  a:=textattr;
[NR5] 1977:  textattr:=col7; normrahmen:=col7;
[UØ4] 1978:  rahmen(2Ø,8,4Ø,7,''); clrscr;
[XF1] 1979:  writeln;
[UH4] 198Ø:  writeln('                SBackup');
[8W4] 1981:  writeln('             Version  1.ØØ');
[XF1] 1982:  writeln;
[9E5] 1983:  writeln('          (c) Copyright 1991');
[VS5] 1984:  write('        DMV Widuch GmbH & Co. KG');
[XF1] 1985:  writeln;
[ØØØ] 1986:
[YE5] 1987:  infoline(' Taste oder Maustaste'+
[EV1] 1988:           ' drücken');
[BM1] 1989:  showmaus;
[RG4] 199Ø:  repeat until keypressed or
[CA3] 1991:         button1 or button2;
[9A4] 1992:  if keypressed then clrbuf;
[LY1] 1993:  infoline('');
[DU4] 1994:  textattr:=a; normrahmen:=a;
[GH2] 1995:  restorescreen;
[MG1] 1996:  nobutton
[ZNØ] 1997:  end;
[ØØØ] 1998:
[2F3] 1999: procedure elpstime;
[BY1] 2ØØØ: var a:byte;
[MUØ] 2Ø01:  begin
[262] 2Ø02:  a:=textattr;
[9L6] 2Ø03:  textattr:=col8; normrahmen:=textattr;
[DC1] 2Ø04:  nomaus;
[F14] 2Ø05:  rahmen(57,5,19,2,''); clrscr;
[FN2] 2Ø06:  tic1:=18; kor1:=5;
[2S4] 2Ø07:  write(' Übertragungsdauer');
[X94] 2Ø08:  gotoxy(7,2); write('Ø.ØØ.ØØ');
[5F1] 2Ø09:  anf:=1Ø98;
```

```
[BM1] 2Ø1Ø:  showmaus;
[1L4] 2Ø11:  textattr:=a; normrahmen:=a
[ZNØ] 2Ø12:  end;
[ØØØ] 2Ø13:
[MUØ] 2Ø14: begin
[LW5] 2Ø15: rec[Ø]:=228; rec[1]:=196; rec[2]:=243;
[PW3] 2Ø16: rec[3]:=171; rec[4]:=235;
[6X5] 2Ø17: for i:=Ø to 4 do endrec[i]:=rec[i];
[MB2] 2Ø18: endrec[5]:=92;
[NN5] 2Ø19: dirno:=1; diskno:=1; lw:=Ø; rlw:=Ø;
[TG5] 2Ø2Ø: zwingflag:=true; verify:=false;
[VS2] 2Ø21: timeflag:=false;
[YN5] 2Ø22: activewindow:=false; fehl:=false;
[UN1] 2Ø23: on:=false;
[MG3] 2Ø24: if maxavail<$ff8Ø then
[MUØ] 2Ø25:   begin
[AE4] 2Ø26:   writeln(' Zu wenig freier'+
[B63] 2Ø27:              ' Speicherplatz!');
[ØD1] 2Ø28:   cursor;
[RPØ] 2Ø29:   halt
[4MØ] 2Ø3Ø:   end
[384] 2Ø31:  else getmem(puffer,$ff8Ø);
[L24] 2Ø32: sflag:=false; nocursor;
[UX5] 2Ø33: a[1]:=' ^Backup '; a[2]:=' ^Restore ';
[WH4] 2Ø34: a[3]:=' B^ackup-Optionen ';
[SM4] 2Ø35: a[4]:=' R^estore-Optionen ';
[PE2] 2Ø36: a[5]:=' ^Quit ';
[7D2] 2Ø37: men[1]:=false;
[R47] 2Ø38: men[2]:=true; men[3]:=true; men[4]:=true;
[FD2] 2Ø39: men[5]:=false;
[UR2] 2Ø4Ø: textattr:=col7;
[ZP3] 2Ø41: selectattribut:=sel1;
[BH2] 2Ø42: hotattr:=hot2;
[KG2] 2Ø43: hotflag:=true;
[NY2] 2Ø44: normrahmen:=col1;
[A32] 2Ø45: save:=false;
[MZ3] 2Ø46: rahmen(1,1,78,22,'SBackup');
[YN3] 2Ø47: save:=true; timeinit;
[8R2] 2Ø48: textattr:=col1;
[DB1] 2Ø49: clrscr;
[D31] 2Ø5Ø: intro;
[W94] 2Ø51: windhline(2,Ø,normrahmen);
[LY1] 2Ø52: infoline('');
[BL5] 2Ø53: if color then winfill(2,4,79,23,'▒',
[RP2] 2Ø54:                 blue shl 4+white)
[4G4] 2Ø55:            else winfill(2,4,79,23,'▓',4);
[WP2] 2Ø56: if not color then
[AU4] 2Ø57:    mausmaske(getmask($f4ff,$74ØØ));
[R34] 2Ø58: aktpath:=orgpath; p1:='';
[YR4] 2Ø59: menumode:=Ø; sidemenu:=true;
[P31] 2Ø6Ø: repeat
[A33] 2Ø61: hormenu(1,5,a,men);
[613] 2Ø62: if not escflag then
[ØR2] 2Ø63:    case menumode of
[GN2] 2Ø64:      1: backuphandler;
[CØ3] 2Ø65:      2: restorehandler;
[NK2] 2Ø66:      3: backohandler;
[3P2] 2Ø67:      4: restohandler;
[Y51] 2Ø68:      5: quit
[ZNØ] 2Ø69:      end;
[MB4] 2Ø7Ø: until menumode=menumode+1
[KNØ] 2Ø71: end.
```

Listing 1. »sbackup.pas« ist das Turbo-Pascal-5.5-Hauptprogramm des Multi-Backup-Systems »SBackup«

```
[9P3]    1: ; Programm: dosadd.asm
[HY5]    2: ; Funktion: Unit für Datei-Handles
[5Y4]    3: ; Sprache:  MASM ab Version 4.Ø
[ZC4]    4: ; Autor:    Frank Emminghaus
[FU4]    5: ; (c)1991 DMV Widuch GmbH & Co. KG
[ØØØ]    6:
[H22]    7: CODE segment
[8U3]    8:  assume cs:CODE, ds:CODE
[ØØØ]    9:
[ØC6]   1Ø: public open, make, closeh, readh, writeh,
[UT4]   11: public getpoi, setpoi, deleteh
[6R6]   12: public readsektor, writesektor, kopiere
[ØØØ]   13:
[9X2]   14: puffer  db 256 dup(Ø)
[ØØØ]   15:
[U42]   16: open proc far
[TB1]   17:  push bp
[PG1]   18:  mov bp,sp
[BC1]   19:  push ds
[8C1]   2Ø:  push cs
[G31]   21:  pop es
[HD3]   22:  mov di,offset puffer
[L92]   23:  lds bx,[bp+12]
[M32]   24:  mov cl,[bx]
[B61]   25:  or cl,cl
[7XØ]   26:  jz op2
[6F1]   27:  xor ch,ch
[CMØ]   28: op1:
[M11]   29:  inc bx
[G32]   3Ø:  mov al,[bx]
[EH2]   31:  mov es:[di],al
[BØ1]   32:  inc di
[FD1]   33:  loop op1
```

```
[GMØ]    34: op2:
[NV3]    35:  mov byte ptr es:[di],Ø
[WA2]    36:  mov ax,[bp+1Ø]
[YE3]    37:  mov dx,offset puffer
[8C1]    38:  push cs
[E31]    39:  pop ds
[WM1]    4Ø:  mov ah,3dh
[381]    41:  int 21h
[H22]    42:  les di,[bp+6]
[481]    43:  jnc op3
[SV3]    44:  mov byte ptr es:[di],1
[372]    45:  jmp short op4
[MMØ]    46: op3:
[NV3]    47:  mov byte ptr es:[di],Ø
[SMØ]    48: op4:
[E31]    49:  pop ds
[131]    5Ø:  pop bp
[NYØ]    51:  ret 1Ø
[VG1]    52: open endp
[ØØØ]    53:
[S32]    54: make proc far
[TB1]    55:  push bp
[PG1]    56:  mov bp,sp
[BC1]    57:  push ds
[8C1]    58:  push cs
[G31]    59:  pop es
[HD3]    6Ø:  mov di,offset puffer
[L92]    61:  lds bx,[bp+12]
[M32]    62:  mov cl,[bx]
[B61]    63:  or cl,cl
[7WØ]    64:  jz ma2
[6F1]    65:  xor ch,ch
[VKØ]    66: ma1:
[M11]    67:  inc bx
[G32]    68:  mov al,[bx]
[EH2]    69:  mov es:[di],al
[BØ1]    7Ø:  inc di
[DB1]    71:  loop ma1
[ZKØ]    72: ma2:
[NV3]    73:  mov byte ptr es:[di],Ø
[ØB2]    74:  mov cx,[bp+1Ø]
[YE3]    75:  mov dx,offset puffer
[8C1]    76:  push cs
[E31]    77:  pop ds
[UM1]    78:  mov ah,3ch
[381]    79:  int 21h
[H22]    8Ø:  les di,[bp+6]
[L61]    81:  jnc ma3
[SV3]    82:  mov byte ptr es:[di],1
[C62]    83:  jmp short ma4
[3LØ]    84: ma3:
[NV3]    85:  mov byte ptr es:[di],Ø
[7LØ]    86: ma4:
[E31]    87:  pop ds
[131]    88:  pop bp
[NYØ]    89:  ret 1Ø
[TF1]    9Ø: make endp
[ØØØ]    91:
[ZL2]    92: closeh proc far
[TB1]    93:  push bp
[PG1]    94:  mov bp,sp
[YA2]    95:  mov bx,[bp+1Ø]
[YM1]    96:  mov ah,3eh
[381]    97:  int 21h
[H22]    98:  les di,[bp+6]
[S61]    99:  jnc cl1
[SV3]   1ØØ:  mov byte ptr es:[di],1
[D81]   1Ø1:  jmp cl2
[8LØ]   1Ø2: cl1:
[NV3]   1Ø3:  mov byte ptr es:[di],Ø
[CLØ]   1Ø4: cl2:
[131]   1Ø5:  pop bp
[FUØ]   1Ø6:  ret 6
[8Y1]   1Ø7: closeh endp
[ØØØ]   1Ø8:
[272]   1Ø9: readh proc far
[TB1]   11Ø:  push bp
[PG1]   111:  mov bp,sp
[UN2]   112:  mov bx, [bp+16]
[8B2]   113:  mov cx,[bp+14]
[BC1]   114:  push ds
[L92]   115:  lds dx,[bp+1Ø]
[ØN1]   116:  mov ah,3fh
[381]   117:  int 21h
[R32]   118:  lds bx,[bp+6]
[171]   119:  jnc rel
[8D3]   12Ø:  mov byte ptr [bx],1
[Ø72]   121:  jmp short re2
[HLØ]   122: rel:
[4D3]   123:  mov byte ptr [bx],Ø
[NLØ]   124: re2:
[E31]   125:  pop ds
[131]   126:  pop bp
[VYØ]   127:  ret 12
[5M1]   128: readh endp
[ØØØ]   129:
[2N2]   13Ø: writeh proc far
[TB1]   131:  push bp
[PG1]   132:  mov bp,sp
[UN2]   133:  mov bx, [bp+16]
```

```
[8B2]   134:  mov cx,[bp+14]
[BC1]   135:  push ds
[L92]   136:  lds dx,[bp+1Ø]
[NL1]   137:  mov ah,4Øh
[381]   138:  int 21h
[R32]   139:  lds bx,[bp+6]
[K81]   14Ø:  jnc wr1
[8D3]   141:  mov byte ptr [bx],1
[182]   142:  jmp short wr2
[2NØ]   143: wr1:
[4D3]   144:  mov byte ptr [bx],Ø
[6NØ]   145: wr2:
[E31]   146:  pop ds
[131]   147:  pop bp
[VYØ]   148:  ret 12
[BZ1]   149: writeh endp
[ØØØ]   15Ø:
[UM2]   151: getpoi proc far
[TB1]   152:  push bp
[PG1]   153:  mov bp,sp
[YA2]   154:  mov bx,[bp+1Ø]
[DV1]   155:  mov ax,42Ø1h
[PH1]   156:  xor cx,cx
[KH1]   157:  mov dx,cx
[381]   158:  int 21h
[H22]   159:  les di,[bp+6]
[8C1]   16Ø:  jnc get1
[SV3]   161:  mov byte ptr es:[di],1
[EC2]   162:  jmp short get2
[KUØ]   163: get1:
[NV3]   164:  mov byte ptr es:[di],Ø
[LUØ]   165: get2:
[131]   166:  pop bp
[FUØ]   167:  ret 6
[3Z1]   168: getpoi endp
[ØØØ]   169:
[KN2]   17Ø: setpoi proc far
[TB1]   171:  push bp
[PG1]   172:  mov bp,sp
[6B2]   173:  mov bx,[bp+14]
[9V1]   174:  mov ax,42ØØh
[2B2]   175:  mov dx,[bp+1Ø]
[4B2]   176:  mov cx,[bp+12]
[381]   177:  int 21h
[H22]   178:  les di,[bp+6]
[ØD1]   179:  jnc set1
[SV3]   18Ø:  mov byte ptr es:[di],1
[YD2]   181:  jmp short set2
[AVØ]   182: set1:
[NV3]   183:  mov byte ptr es:[di],Ø
[BVØ]   184: set2:
[131]   185:  pop bp
[NYØ]   186:  ret 1Ø
[VZ1]   187: setpoi endp
[ØØØ]   188:
[ØS2]   189: deleteh proc far
[TB1]   19Ø:  push bp
[PG1]   191:  mov bp,sp
[BC1]   192:  push ds
[8C1]   193:  push cs
[G31]   194:  pop es
[HD3]   195:  mov di,offset puffer
[F92]   196:  lds bx,[bp+1Ø]
[M32]   197:  mov cl,[bx]
[B61]   198:  or cl,cl
[W41]   199:  jz del2
[6F1]   2ØØ:  xor ch,ch
[CTØ]   2Ø1: del1:
[M11]   2Ø2:  inc bx
[G32]   2Ø3:  mov al,[bx]
[EH2]   2Ø4:  mov es:[di],al
[BØ1]   2Ø5:  inc di
[ZE1]   2Ø6:  loop del1
[DTØ]   2Ø7: del2:
[NV3]   2Ø8:  mov byte ptr es:[di],Ø
[WA2]   2Ø9:  mov ax,[bp+1Ø]
[YE3]   21Ø:  mov dx,offset puffer
[8C1]   211:  push cs
[E31]   212:  pop ds
[RL1]   213:  mov ah,41h
[381]   214:  int 21h
[H22]   215:  les di,[bp+6]
[4B1]   216:  jnc del3
[SV3]   217:  mov byte ptr es:[di],1
[SB2]   218:  jmp short del4
[ETØ]   219: del3:
[NV3]   22Ø:  mov byte ptr es:[di],Ø
[FTØ]   221: del4:
[E31]   222:  pop ds
[131]   223:  pop bp
[LUØ]   224:  ret 8
[R22]   225: deleteh endp
[ØØØ]   226:
[8B3]   227: readsektor proc far
[TB1]   228:  push bp
[PG1]   229:  mov bp,sp
[8A2]   23Ø:  mov al,[bp+18]
[EB2]   231:  mov dx,[bp+16]
```

Listing 2. Das Assembler-Programm »dosadd.asm« stellt Maschinenroutinen für die Dateiein- und -ausgabe

```
[8B2]  232:  mov cx,[bp+14]
[BC1]  233:  push ds
[F92]  234:  lds bx,[bp+1Ø]
[3B1]  235:  push di
[TB1]  236:  push bp
[F81]  237:  int 25h
[R31]  238:  pop ax
[131]  239:  pop bp
[G21]  24Ø:  pop di
[R32]  241:  lds bx,[bp+6]
[X51]  242:  jc read1
[4D3]  243:  mov byte ptr [bx],Ø
[KM2]  244:  jmp short read2
[BYØ]  245: read1:
[8D3]  246:  mov byte ptr [bx],1
[DYØ]  247: read2:
[E31]  248:  pop ds
[7H1]  249:  mov sp,bp
[131]  25Ø:  pop bp
[1ZØ]  251:  ret 14
[HN2]  252: readsektor endp
[ØØØ]  253:
[SL3]  254: writesektor proc far
[TB1]  255:  push bp
[PG1]  256:  mov bp,sp
[8A2]  257:  mov al,[bp+18]
[EB2]  258:  mov dx,[bp+16]
[8B2]  259:  mov cx,[bp+14]
[BC1]  26Ø:  push ds
[F92]  261:  lds bx,[bp+1Ø]
[3B1]  262:  push di
[TB1]  263:  push bp
[K81]  264:  int 26h
[R31]  265:  pop ax
[131]  266:  pop bp
[G21]  267:  pop di
[R32]  268:  lds bx,[bp+6]
[AD1]  269:  jc write1
[4D3]  27Ø:  mov byte ptr [bx],Ø
[HV2]  271:  jmp short write2
[191]  272: write1:
[8D3]  273:  mov byte ptr [bx],1
[491]  274: write2:
[E31]  275:  pop ds
[7H1]  276:  mov sp,bp
[131]  277:  pop bp
[1ZØ]  278:  ret 14
[FY2]  279: writesektor endp
[ØØØ]  28Ø:
[XT2]  281: kopiere proc far
[TB1]  282:  push bp
[PG1]  283:  mov bp,sp
[ULØ]  284:  cld
[BC1]  285:  push ds
[852]  286:  mov cx,[bp+6]
[E32]  287:  lds si,[bp+8]
[D82]  288:  les di,[bp+12]
[9M1]  289:  rep movsb
[E31]  29Ø:  pop ds
[7H1]  291:  mov sp,bp
[131]  292:  pop bp
[NYØ]  293:  ret 1Ø
[M42]  294: kopiere endp
[ØØØ]  295:
[TF1]  296: CODE ends
[4MØ]  297:  end
```

Listing 2. Das Assembler-Programm »dosadd.asm« stellt Maschinenroutinen für die Dateiein- und -ausgabe

```
[9D3]    1: ; Programm: maus.asm
[U37]    2: ; Funktion: Maustreiber-Grundfunktionen
[5Y4]    3: ; Sprache:  MASM ab Version 4.Ø
[ZC4]    4: ; Autor:    Frank Emminghaus
[FU4]    5: ; (C)1991 DMV Widuch GmbH & Co. KG
[ØØØ]    6:
[444]    7: DATA segment word public
[DG1]    8: DATA ends
[ØØØ]    9:
[X24]   1Ø: CODE segment byte public
[XE2]   11:     assume cs: CODE
[ØØØ]   12:
[UD3]   13: public firsthandler
[ØØØ]   14:
[V14]   15: extrn maushandler: near
[ØØØ]   16:
[GK3]   17: firsthandler proc far
[PC1]   18:   push ax
[TC1]   19:   push bx
[WC1]   2Ø:   push cx
[ZC1]   21:   push dx
[3B1]   22:   push di
[GC1]   23:   push si
[TB1]   24:   push bp
[EC1]   25:   push es
[BC1]   26:   push ds
[ØØØ]   27:
[4M5]   28:   push ax          ; Parameter auf Stack
[PM1]   29:                    ; Ereignis
```

```
[RH4]   3Ø:   push bx          ; ButtonStatus
[UX5]   31:   push cx          ; horizontale Position
[WT5]   32:   push dx          ; vertikale Posiztion
[ØØØ]   33:
[N12]   34:   mov ax, DATA
[1H1]   35:   mov ds,ax
[7R2]   36:   call maushandler
[ØØØ]   37:
[E31]   38:   pop ds
[G31]   39:   pop es
[131]   4Ø:   pop bp
[E31]   41:   pop si
[G21]   42:   pop di
[X31]   43:   pop dx
[V31]   44:   pop cx
[T31]   45:   pop bx
[R31]   46:   pop ax
[ØØØ]   47:
[3PØ]   48:   ret
[HY2]   49: firsthandler endp
[ØØØ]   5Ø:
[TF1]   51: CODE ends
[4MØ]   52:  end
```

Listing 3. Die Maustreiber-Grundlagen sind in »maus.asm« enthalten

```
[LS3]    1: ; Programm: neuint.asm
[ZY6]    2: ; Funktion: Einlage für Timerinterrupt
[5Y4]    3: ; Sprache:  MASM ab Version 4.Ø
[ZC4]    4: ; Autor:    Frank Emminghaus
[FU4]    5: ; (C)1991 DMV Widuch GmbH & Co. KG
[ØØØ]    6:
[444]    7: DATA segment word public
[283]    8:  extrn oldint:dword
[NE2]    9:  extrn tic:byte
[EP2]   1Ø:  extrn tic1:byte
[NE2]   11:  extrn kor:byte
[EP2]   12:  extrn kor1:byte
[CT2]   13:  extrn secs:byte
[P92]   14:  extrn on:byte
[8P2]   15:  extrn bild:word
[RE2]   16:  extrn anf:word
[DG1]   17: DATA ends
[ØØØ]   18:
[X24]   19: CODE segment byte public
[DU3]   2Ø:   assume cs:CODE, ds:DATA
[ØØØ]   21:
[Z52]   22: public neuint
[ØØØ]   23:
[4P2]   24: neuint proc far
[PC1]   25:  push ax
[TC1]   26:  push bx
[EC1]   27:  push es
[BC1]   28:  push ds
[ZZ1]   29:  mov ax,DATA
[1H1]   3Ø:  mov ds,ax
[ØØØ]   31:
[K81]   32:  dec tic
[7A1]   33:  jnz we1
[4P1]   34:  mov tic,18
[EA1]   35:  dec kor
[BA1]   36:  jnz we2
[ØL1]   37:  mov kor,5
[R91]   38:  inc tic
[ZLØ]   39: we2:
[BC1]   4Ø:  push ds
[G31]   41:  pop es
[SZ1]   42:  mov ds,bild
[CL1]   43:  mov bx,152
[PE3]   44:  cmp byte ptr [bx],':'
[FA1]   45:  jnz we3
[EF3]   46:  mov byte ptr [bx],' '
[S72]   47:  jmp short we4
[3MØ]   48: we3:
[8G3]   49:  mov byte ptr [bx],':'
[7MØ]   5Ø: we4:
[ØZ1]   51:  dec es:secs
[RA1]   52:  jnz we5
[4C2]   53:  mov es:secs,6Ø
[SL1]   54:  mov bx,156
[9Ø3]   55:  inc byte ptr [bx]
[NE3]   56:  cmp byte ptr [bx],'9'
[L71]   57:  jna we5
[YF3]   58:  mov byte ptr [bx],'Ø'
[R81]   59:  sub bl,2
[9Ø3]   6Ø:  inc byte ptr [bx]
[KE3]   61:  cmp byte ptr [bx],'6'
[RA1]   62:  jnz we5
[YF3]   63:  mov byte ptr [bx],'Ø'
[T81]   64:  sub bl,4
[9Ø3]   65:  inc byte ptr [bx]
[GE3]   66:  cmp byte ptr [bx],'4'
[VA1]   67:  jnz we6
[4W3]   68:  cmp byte ptr [bx-2],'2'
[VA1]   69:  jnz we6
[YF3]   7Ø:  mov byte ptr [bx],'Ø'
[GX3]   71:  mov byte ptr [bx-2],'Ø'
```

```
[U72]    72:  jmp short we5
[FMØ]    73: we6:
[NE3]    74:  cmp byte ptr [bx],'9'
[L71]    75:  jna we5
[YF3]    76:  mov byte ptr [bx],'Ø'
[R81]    77:  sub bl,2
[9Ø3]    78:  inc byte ptr [bx]
[ØØØ]    79:
[BMØ]    8Ø: we5:
[EC1]    81:  push es
[E31]    82:  pop ds
[VLØ]    83: we1:
[N91]    84:  cmp on,Ø
[8XØ]    85:  jz we7
[3A1]    86:  dec tic1
[ZA1]    87:  jnz we7
[AW1]    88:  mov tic1,18
[ZB1]    89:  dec kor1
[3B1]    9Ø:  jnz we8
[KR1]    91:  mov kor1,5
[8B1]    92:  inc tic1
[RMØ]    93: we8:
[AR1]    94:  mov bx,anf
[BC1]    95:  push ds
[SZ1]    96:  mov ds,bild
[9Ø3]    97:  inc byte ptr [bx]
[NE3]    98:  cmp byte ptr [bx],'9'
[381]    99:  jna we9
[YF3]   1ØØ:  mov byte ptr [bx],'Ø'
[V91]   1Ø1:  sub bx,2
[9Ø3]   1Ø2:  inc byte ptr [bx]
[KE3]   1Ø3:  cmp byte ptr [bx],'6'
[7B1]   1Ø4:  jnz we9
[YF3]   1Ø5:  mov byte ptr [bx],'Ø'
[X91]   1Ø6:  sub bx,4
[9Ø3]   1Ø7:  inc byte ptr [bx]
[NE3]   1Ø8:  cmp byte ptr [bx],'9'
[381]   1Ø9:  jna we9
[YF3]   11Ø:  mov byte ptr [bx],'Ø'
[V91]   111:  sub bx,2
[9Ø3]   112:  inc byte ptr [bx]
[KE3]   113:  cmp byte ptr [bx],'6'
[7B1]   114:  jnz we9
[YF3]   115:  mov byte ptr [bx],'Ø'
[TD3]   116:  inc byte ptr [bx-4]
[FX1]   117: we9: pop ds
[CP1]   118: we7: pushf
[7A2]   119:      call [oldint]
[ØØØ]   12Ø:
[E31]   121:      pop ds
[G31]   122:      pop es
[T31]   123:      pop bx
[R31]   124:      pop ax
[GRØ]   125:      iret
[DØ2]   126: neuint endp
[ØØØ]   127:
[TF1]   128: CODE ends
[4MØ]   129:  end
```

Listing 4. »neuint.asm« enthält die eingehängte Routine für den Timerinterrupt

```
[U54]     1: ; Programm: strings.asm
[8M6]     2: ; Funktion: Stringbehandlungsroutinen
[5Y4]     3: ; Sprache:  MASM ab Version 4.Ø
[ZC4]     4: ; Autor:    Frank Emminghaus
[FU4]     5: ; (c)1991 DMV Widuch GmbH & Co. KG
[ØØØ]     6:
[H22]     7: CODE segment
[EA2]     8: assume cs:CODE
[GZ5]     9: public asstring, astype, left, right
[4X6]    1Ø: public upchar, upstring, lochar, lostring
[3T6]    11: public prefix,suffix, substring, intval
[ØT6]    12: public wordval, delleft, delright, intstr
[V76]    13: public alstyp, wordstr, scanleft, copy
[3T6]    14: public scanright, vorkommen, getposfeld
[ØØØ]    15:
[EL2]    16: copys proc near
[TB1]    17:  push bp
[PG1]    18:  mov bp,sp
[BC1]    19:  push ds
[ULØ]    2Ø:  cld
[6F1]    21:  xor ch,ch
[F22]    22:  les di,[bp+4]
[E32]    23:  lds si,[bp+8]
[CVØ]    24:  lodsb
[KF1]    25:  mov cl,al
[NXØ]    26:  stosb
[9M1]    27:  rep movsb
[E31]    28:  pop ds
[7H1]    29:  mov sp,bp
[131]    3Ø:  pop bp
[LUØ]    31:  ret 8
[CR1]    32: copys endp
[ØØØ]    33:
[3Y2]    34: asstring proc far
[TB1]    35:  push bp
[PG1]    36:  mov bp,sp
[GB2]    37:  les di, [bp+6]
```

```
[EC1]    38:  push es
[3B1]    39:  push di
[982]    4Ø:  les di,[bp+1Ø]
[EC1]    41:  push es
[3B1]    42:  push di
[5S1]    43:  call copys
[7H1]    44:  mov sp,bp
[131]    45:  pop bp
[BUØ]    46:  ret 4
[4A2]    47: asstring endp
[ØØØ]    48:
[RN2]    49: astype proc far
[TB1]    5Ø:  push bp
[PG1]    51:  mov bp,sp
[S32]    52:  les ax,[bp+6]
[AH1]    53:  mov dx,es
[131]    54:  pop bp
[BUØ]    55:  ret 4
[ØØ2]    56: astype endp
[ØØØ]    57:
[X32]    58: left proc far
[TB1]    59:  push bp
[PG1]    6Ø:  mov bp,sp
[S61]    61:  add bp,6
[Ø42]    62:  mov cx,[bp]
[BC1]    63:  push ds
[832]    64:  lds si,[bp+2]
[H22]    65:  les di,[bp+6]
[DG2]    66:  mov ch,ds:[si]
[PD1]    67:  cmp ch,cl
[HYØ]    68:  jnb l1
[3N1]    69:  xchg cl,ch
[6F1]    7Ø:  xor ch,ch
[T42]    71:  mov [bp],cx
[LGØ]    72: l1:
[ULØ]    73:  cld
[6F1]    74:  xor ch,ch
[P11]    75:  inc cx
[9M1]    76:  rep movsb
[Ø42]    77:  mov cx,[bp]
[R32]    78:  lds bx,[bp+6]
[942]    79:  mov [bx],cl
[E31]    8Ø:  pop ds
[131]    81:  pop bp
[FUØ]    82:  ret 6
[VV2]    83:  string db 256 dup(Ø)
[YF1]    84: left endp
[ØØØ]    85:
[292]    86: right proc far
[TB1]    87:  push bp
[PG1]    88:  mov bp,sp
[S61]    89:  add bp,6
[KNØ]    9Ø:  std
[Ø42]    91:  mov cx,[bp]
[6F1]    92:  xor ch,ch
[BC1]    93:  push ds
[832]    94:  lds si,[bp+2]
[H22]    95:  les di,[bp+6]
[NG2]    96:  mov al,ds:[si]
[ØF1]    97:  xor ah,ah
[UF1]    98:  cmp ax,cx
[XYØ]    99:  jnb r1
[EH1]   1ØØ:  mov cx,ax
[T42]   1Ø1:  mov [bp],cx
[ZGØ]   1Ø2: r1:
[K11]   1Ø3:  inc ax
[P11]   1Ø4:  inc cx
[ØD1]   1Ø5:  add si,ax
[4C1]   1Ø6:  add di,cx
[9M1]   1Ø7:  rep movsb
[Ø42]   1Ø8:  mov cx,[bp]
[R32]   1Ø9:  lds bx,[bp+6]
[942]   11Ø:  mov [bx],cl
[E31]   111:  pop ds
[131]   112:  pop bp
[FUØ]   113:  ret 6
[5P1]   114: right endp
[ØØØ]   115:
[AN2]   116: upcase proc near
[AC1]   117:  cmp al,97
[ZMØ]   118:  jb u8
[EG1]   119:  cmp al,122
[V91]   12Ø:  ja usond
[DK1]   121:  sub al,2Øh
[V32]   122:  jmp short u8
[331]   123: usond:
[3H1]   124:  cmp al,129
[321]   125:  jnz u1
[CK1]   126:  mov al,154
[V32]   127:  jmp short u8
[N22]   128: u1: cmp al,132
[621]   129:  jnz u2
[4K1]   13Ø:  mov al,142
[V32]   131:  jmp short u8
[C32]   132: u2: cmp al,148
[921]   133:  jnz u3
[9K1]   134:  mov al,153
[V32]   135:  jmp short u8
```

Listing 5. »strings.asm« beinhaltet die String-Behandlungsroutinen

```
[N22]  136: u3: cmp al,13Ø
[C21]  137:  jnz u4
[AK1]  138:  mov al,144
[V32]  139:  jmp short u8
[732]  14Ø: u4: cmp al,135
[F21]  141:  jnz u5
[KK1]  142:  mov al,128
[V32]  143:  jmp short u8
[732]  144: u5: cmp al,134
[K21]  145:  jnz u6
[7K1]  146:  mov al,143
[V32]  147:  jmp short u8
[G32]  148: u6: cmp al,164
[N21]  149:  jnz u7
[HK1]  15Ø:  mov al,165
[V32]  151:  jmp short u8
[K32]  152: u7: cmp al,145
[S21]  153:  jnz u8
[GK1]  154:  mov al,146
[X81]  155: u8: ret
[9Y1]  156: upcase endp
[ØØØ]  157:
[NM2]  158: locase proc near
[3C1]  159:  cmp al,65
[T71]  16Ø:  jb raus
[WB1]  161:  cmp al,9Ø
[CE1]  162:  ja losond
[YF1]  163:  add al,2Øh
[VG2]  164:  jmp short raus
[GB1]  165: losond:
[UG1]  166:  cmp al,154
[FA1]  167:  jnz lo1
[NK1]  168:  mov al,129
[VG2]  169:  jmp short raus
[N92]  17Ø: lo1: cmp al,142
[LA1]  171:  jnz lo2
[2K1]  172:  mov al,132
[VG2]  173:  jmp short raus
[ØA2]  174: lo2: cmp al,153
[RA1]  175:  jnz lo3
[PK1]  176:  mov al,148
[VG2]  177:  jmp short raus
[FA2]  178: lo3: cmp al,165
[VA1]  179:  jnz lo4
[EK1]  18Ø:  mov al,164
[VG2]  181:  jmp short raus
[5A2]  182: lo4: cmp al,143
[ZA1]  183:  jnz lo5
[8K1]  184:  mov al,134
[VG2]  185:  jmp short raus
[NA2]  186: lo5: cmp al,146
[3B1]  187:  jnz lo6
[DK1]  188:  mov al,145
[VG2]  189:  jmp short raus
[HA2]  19Ø: lo6: cmp al,144
[7B1]  191:  jnz lo7
[WH1]  192:  mov al,13Ø
[VG2]  193:  jmp short raus
[ZA2]  194: lo7: cmp al,128
[MG1]  195:  jnz raus
[BK1]  196:  mov al,135
[RG1]  197: raus: ret
[MX1]  198: locase endp
[ØØØ]  199:
[AM2]  2ØØ: upchar proc far
[TB1]  2Ø1:  push bp
[PG1]  2Ø2:  mov bp,sp
[452]  2Ø3:  mov ax,[bp+6]
[PØ2]  2Ø4:  call upcase
[131]  2Ø5:  pop bp
[7UØ]  2Ø6:  ret 2
[LY1]  2Ø7: upchar endp
[ØØØ]  2Ø8:
[NL2]  2Ø9: lochar proc far
[TB1]  21Ø:  push bp
[PG1]  211:  mov bp,sp
[452]  212:  mov ax,[bp+6]
[RZ1]  213:  call locase
[131]  214:  pop bp
[7UØ]  215:  ret 2
[YX1]  216: lochar endp
[ØØØ]  217:
[2Z2]  218: upstring proc far
[TB1]  219:  push bp
[PG1]  22Ø:  mov bp,sp
[S61]  221:  add bp,6
[BC1]  222:  push ds
[F22]  223:  lds bx,[bp]
[F22]  224:  les di,[bp+4]
[M32]  225:  mov cl,[bx]
[KH2]  226:  mov es:[di],cl
[6F1]  227:  xor ch,ch
[B61]  228:  or cl,cl
[FG1]  229:  jz fertig
[ZRØ]  23Ø: lop:
[M11]  231:  inc bx
[BØ1]  232:  inc di
[G32]  233:  mov al,[bx]
[PØ2]  234:  call upcase
[EH2]  235:  mov es:[di],al
```

```
[1E1]  236:  loop lop
[6A1]  237: fertig:
[E31]  238:  pop ds
[131]  239:  pop bp
[BUØ]  24Ø:  ret 4
[3B2]  241: upstring endp
[ØØØ]  242:
[DY2]  243: lostring proc far
[TB1]  244:  push bp
[PG1]  245:  mov bp,sp
[S61]  246:  add bp,6
[BC1]  247:  push ds
[F22]  248:  lds bx,[bp]
[F22]  249:  les di,[bp+4]
[M32]  25Ø:  mov cl,[bx]
[KH2]  251:  mov es:[di],cl
[6F1]  252:  xor ch,ch
[B61]  253:  or cl,cl
[2N1]  254:  jz fertig1
[AVØ]  255: lop1:
[M11]  256:  inc bx
[BØ1]  257:  inc di
[G32]  258:  mov al,[bx]
[RZ1]  259:  call locase
[EH2]  26Ø:  mov es:[di],al
[3H1]  261:  loop lop1
[WA1]  262: fertig1:
[E31]  263:  pop ds
[131]  264:  pop bp
[BUØ]  265:  ret 4
[EA2]  266: lostring endp
[ØØØ]  267:
[EN2]  268: prefix proc far
[TB1]  269:  push bp
[PG1]  27Ø:  mov bp,sp
[S61]  271:  add bp,6
[BC1]  272:  push ds
[F22]  273:  lds bx,[bp]
[F22]  274:  les di,[bp+4]
[M32]  275:  mov cl,[bx]
[ZF2]  276:  mov ch,es:[di]
[B61]  277:  or cl,cl
[9D1]  278:  jz ptrue
[PD1]  279:  cmp ch,cl
[6C1]  28Ø:  jb pfalse
[6F1]  281:  xor ch,ch
[4WØ]  282: plop:
[M11]  283:  inc bx
[BØ1]  284:  inc di
[G32]  285:  mov al,[bx]
[NE2]  286:  cmp al,es:[di]
[XT1]  287:  jnz pfalse
[ØL1]  288:  loop plop
[H31]  289: ptrue:
[AB1]  29Ø:  mov ax,1
[9XØ]  291: pout:
[E31]  292:  pop ds
[131]  293:  pop bp
[LUØ]  294:  ret 8
[Z81]  295: pfalse:
[GH1]  296:  xor ax,ax
[GG1]  297:  jmp pout
[RZ1]  298: prefix endp
[ØØØ]  299:
[1P2]  3ØØ: suffix proc far
[TB1]  3Ø1:  push bp
[PG1]  3Ø2:  mov bp,sp
[S61]  3Ø3:  add bp,6
[BC1]  3Ø4:  push ds
[F22]  3Ø5:  lds bx,[bp]
[F22]  3Ø6:  les di,[bp+4]
[M32]  3Ø7:  mov cl,[bx]
[DG2]  3Ø8:  mov dl,es:[di]
[B61]  3Ø9:  or cl,cl
[CD1]  31Ø:  jz strue
[4E1]  311:  cmp dl,cl
[9C1]  312:  jb sfalse
[9F1]  313:  xor dh,dh
[6F1]  314:  xor ch,ch
[5C1]  315:  add di,dx
[DD1]  316:  add bx,cx
[AWØ]  317: slop:
[G32]  318:  mov al,[bx]
[NE2]  319:  cmp al,es:[di]
[3U1]  32Ø:  jnz sfalse
[FØ1]  321:  dec bx
[6ZØ]  322:  dec di
[9L1]  323:  loop slop
[R31]  324: strue:
[AB1]  325:  mov ax,1
[FXØ]  326: sout:
[E31]  327:  pop ds
[131]  328:  pop bp
[LUØ]  329:  ret 8
[591]  33Ø: sfalse:
[GH1]  331:  xor ax,ax
[PG1]  332:  jmp sout
[AØ2]  333: suffix endp
[ØØØ]  334:
[ØØØ]  335:
```

```
[N13]  336: substring proc far
[TB1]  337:  push bp
[PG1]  338:  mov bp,sp
[S61]  339:  add bp,6
[BC1]  34Ø:  push ds
[F22]  341:  lds bx,[bp]
[F22]  342:  les di,[bp+4]
[M32]  343:  mov cl,[bx]
[DG2]  344:  mov dl,es:[di]
[6F1]  345:  xor ch,ch
[2F1]  346:  mov dh,ch
[WE1]  347:  mov ah,ch
[B61]  348:  or cl,cl
[3Z1]  349:  jz subfalse
[M11]  35Ø:  inc bx
[KUØ]  351: sub1:
[BØ1]  352:  inc di
[4E1]  353:  cmp dl,cl
[VW1]  354:  jb subfalse
[FZØ]  355:  dec dl
[EY1]  356:  js subfalse
[3B1]  357:  push di
[TC1]  358:  push bx
[WC1]  359:  push cx
[FB1]  36Ø: sublop:
[G32]  361:  mov al,[bx]
[NE2]  362:  cmp al,es:[di]
[8D2]  363:  jnz subweiter
[BØ1]  364:  inc di
[M11]  365:  inc bx
[552]  366:  loop sublop
[LUØ]  367: sub2:
[V31]  368:  pop cx
[T31]  369:  pop bx
[G21]  37Ø:  pop di
[KF1]  371: subtrue:
[FG1]  372:  mov ax,di
[G32]  373:  sub ax,[bp+4]
[AC1]  374: subout:
[E31]  375:  pop ds
[131]  376:  pop bp
[LUØ]  377:  ret 8
[GU1]  378: subweiter:
[V31]  379:  pop cx
[T31]  38Ø:  pop bx
[G21]  381:  pop di
[TD1]  382:  jmp sub1
[YK1]  383: subfalse:
[GH1]  384:  xor ax,ax
[XV1]  385:  jmp subout
[SE2]  386: substring endp
[ØØØ]  387:
[CN2]  388: intval proc far
[TB1]  389:  push bp
[PG1]  39Ø:  mov bp,sp
[S61]  391:  add bp,6
[912]  392:  les di,[bp]
[BC1]  393:  push ds
[N32]  394:  lds bx,[bp+4]
[M32]  395:  mov cl,[bx]
[6F1]  396:  xor ch,ch
[EF1]  397:  mov dl,ch
[HVØ]  398: int1:
[M11]  399:  inc bx
[G32]  4ØØ:  mov al,[bx]
[TE1]  4Ø1:  cmp al,'-'
[4G1]  4Ø2:  jnz int2
[KA1]  4Ø3:  xor dl,1
[UE1]  4Ø4:  jmp int3
[KVØ]  4Ø5: int2:
[NE1]  4Ø6:  cmp al,'+'
[6G1]  4Ø7:  jnz int4
[LVØ]  4Ø8: int3:
[UG1]  4Ø9:  loop int1
[GH1]  41Ø:  xor ax,ax
[WC2]  411:  jmp intweiter
[MVØ]  412: int4:
[7C1]  413:  cmp al,48
[KZ1]  414:  jb intfehler
[6C1]  415:  cmp al,57
[FZ1]  416:  ja intfehler
[ZE1]  417:  cmp al,'Ø'
[RN1]  418:  jnz intok
[M11]  419:  inc bx
[G32]  42Ø:  mov al,[bx]
[ØH1]  421:  loop int4
[GH1]  422:  xor ax,ax
[WC2]  423:  jmp intweiter
[Z21]  424: intok:
[DD1]  425:  add bx,cx
[FØ1]  426:  dec bx
[W81]  427:  cmp cl,5
[FZ1]  428:  ja intfehler
[ØØØ]  429:
[ZC1]  43Ø:  push dx
[1B1]  431:  mov si,1
[GH1]  432:  xor ax,ax
[PC1]  433:  push ax
[NVØ]  434: int5:
[G32]  435:  mov al,[bx]
```

```
[7C1]  436:  cmp al,48
[ZN2]  437:  jb intfehlerpop
[6C1]  438:  cmp al,57
[WN2]  439:  ja intfehlerpop
[EK1]  44Ø:  sub al,3Øh
[ØF1]  441:  xor ah,ah
[A31]  442:  mul si
[M71]  443:  or dx,dx
[FØ3]  444:  jnz intfehlerpop
[X31]  445:  pop dx
[CD1]  446:  add ax,dx
[R22]  447:  jc intfehler1
[PC1]  448:  push ax
[AE1]  449:  mov ax,1Ø
[A31]  45Ø:  mul si
[1H1]  451:  mov si,ax
[FØ1]  452:  dec bx
[2H1]  453:  loop int5
[R31]  454:  pop ax
[X31]  455:  pop dx
[G61]  456:  or dl,dl
[V61]  457:  jz int6
[RT1]  458:  cmp ax,8ØØØh
[FZ1]  459:  ja intfehler
[H11]  46Ø:  neg ax
[WC2]  461:  jmp intweiter
[PVØ]  462: int6:
[612]  463:  test ax,8ØØØh
[LC2]  464:  jnz intfehler
[FV1]  465: intweiter:
[NV3]  466:  mov byte ptr es:[di],Ø
[9D1]  467: intout:
[E31]  468:  pop ds
[131]  469:  pop bp
[LUØ]  47Ø:  ret 8
[XB2]  471: intfehlerpop:
[R31]  472:  pop ax
[8Ø2]  473: intfehler1:
[R31]  474:  pop ax
[VT1]  475: intfehler:
[SV3]  476:  mov byte ptr es:[di],1
[GH1]  477:  xor ax,ax
[WW1]  478:  jmp intout
[NZ1]  479: intval endp
[ØØØ]  48Ø:
[3V2]  481: wordval proc far
[TB1]  482:  push bp
[PG1]  483:  mov bp,sp
[S61]  484:  add bp,6
[912]  485:  les di,[bp]
[BC1]  486:  push ds
[N32]  487:  lds bx,[bp+4]
[M32]  488:  mov cl,[bx]
[6F1]  489:  xor ch,ch
[EF1]  49Ø:  mov dl,ch
[R11]  491: word1:
[M11]  492:  inc bx
[G32]  493:  mov al,[bx]
[TE1]  494:  cmp al,'-'
[HM1]  495:  jnz word2
[KA1]  496:  xor dl,1
[8L1]  497:  jmp word3
[T11]  498: word2:
[NE1]  499:  cmp al,'+'
[NM1]  5ØØ:  jnz word4
[V11]  5Ø1: word3:
[ZR1]  5Ø2:  loop word1
[GH1]  5Ø3:  xor ax,ax
[RL2]  5Ø4:  jmp wordweiter
[X11]  5Ø5: word4:
[7C1]  5Ø6:  cmp al,48
[P42]  5Ø7:  jb wordfehler
[6C1]  5Ø8:  cmp al,57
[L42]  5Ø9:  ja wordfehler
[ZE1]  51Ø:  cmp al,'Ø'
[CX1]  511:  jnz wordok
[M11]  512:  inc bx
[G32]  513:  mov al,[bx]
[8S1]  514:  loop word4
[GH1]  515:  xor ax,ax
[RL2]  516:  jmp wordweiter
[EC1]  517: wordok:
[DD1]  518:  add bx,cx
[FØ1]  519:  dec bx
[W81]  52Ø:  cmp cl,5
[L42]  521:  ja wordfehler
[ØØØ]  522:
[ZC1]  523:  push dx
[1B1]  524:  mov si,1
[GH1]  525:  xor ax,ax
[PC1]  526:  push ax
[Z11]  527: word5:
[G32]  528:  mov al,[bx]
[7C1]  529:  cmp al,48
[KS2]  53Ø:  jb wordfehlerpop
[6C1]  531:  cmp al,57
[FS2]  532:  ja wordfehlerpop
```

Listing 5. »strings.asm« beinhaltet die String-Behandlungsroutinen

```
[EK1]  533:  sub al,3Øh
[ØF1]  534:  xor ah,ah
[A31]  535:  mul si
[M71]  536:  or dx,dx
[353]  537:  jnz wordfehlerpop
[X31]  538:  pop dx
[CD1]  539:  add ax,dx
[C92]  54Ø:  jc wordfehler1
[PC1]  541:  push ax
[AE1]  542:  mov ax,1Ø
[A31]  543:  mul si
[1H1]  544:  mov si,ax
[FØ1]  545:  dec bx
[BS1]  546:  loop word5
[R31]  547:  pop ax
[X31]  548:  pop dx
[G61]  549:  or dl,dl
[ML2]  55Ø:  jnz wordfehler
[442]  551: wordweiter:
[NV3]  552:  mov byte ptr es:[di],Ø
[EK1]  553: wordout:
[E31]  554:  pop ds
[131]  555:  pop bp
[LUØ]  556:  ret 8
[BK2]  557: wordfehlerpop:
[R31]  558:  pop ax
[C32]  559: wordfehler1:
[R31]  56Ø:  pop ax
[P22]  561: wordfehler:
[SV3]  562:  mov byte ptr es:[di],1
[GH1]  563:  xor ax,ax
[F72]  564:  jmp wordout
[U52]  565: wordval endp
[ØØØ]  566:
[WS2]  567: delleft proc far
[TB1]  568:  push bp
[PG1]  569:  mov bp,sp
[S61]  57Ø:  add bp,6
[242]  571:  mov dx,[bp]
[BC1]  572:  push ds
[L32]  573:  lds bx,[bp+2]
[M32]  574:  mov cl,[bx]
[3E1]  575:  cmp cl,dl
[YA1]  576:  jnb del1
[PF1]  577:  mov dl,cl
[CTØ]  578: del1:
[9F1]  579:  xor dh,dh
[M11]  58Ø:  inc bx
[4G1]  581:  mov di,bx
[5C1]  582:  add di,dx
[3E1]  583:  sub cl,dl
[PF1]  584:  mov dl,cl
[2P1]  585:  jz machnix
[6F1]  586:  xor ch,ch
[DTØ]  587: del2:
[722]  588:  mov al,[di]
[342]  589:  mov [bx],al
[BØ1]  59Ø:  inc di
[M11]  591:  inc bx
[1F1]  592:  loop del2
[2E1]  593: machnix:
[252]  594:  mov bx,[bp+2]
[C42]  595:  mov [bx],dl
[E31]  596:  pop ds
[131]  597:  pop bp
[FUØ]  598:  ret 6
[L32]  599: delleft endp
[ØØØ]  6ØØ:
[YU2]  6Ø1: delright proc far
[TB1]  6Ø2:  push bp
[PG1]  6Ø3:  mov bp,sp
[S61]  6Ø4:  add bp,6
[242]  6Ø5:  mov dx,[bp]
[BC1]  6Ø6:  push ds
[L32]  6Ø7:  lds bx,[bp+2]
[M32]  6Ø8:  mov cl,[bx]
[3E1]  6Ø9:  cmp cl,dl
[4P1]  61Ø:  jnb right1
[PF1]  611:  mov dl,cl
[681]  612: right1:
[3E1]  613:  sub cl,dl
[942]  614:  mov [bx],cl
[E31]  615:  pop ds
[131]  616:  pop bp
[FUØ]  617:  ret 6
[Z62]  618: delright endp
[ØØØ]  619:
[1R2]  62Ø: intstr proc far
[TB1]  621:  push bp
[PG1]  622:  mov bp,sp
[452]  623:  mov ax,[bp+6]
[BC1]  624:  push ds
[8C1]  625:  push cs
[E31]  626:  pop ds
[US3]  627:  mov bx,offset string+7
[TH1]  628:  xor dx,dx
[HH1]  629:  mov cx,dx
[1E1]  63Ø:  mov si,1Ø
[612]  631:  test ax,8ØØØh
[UWØ]  632:  jz in1
```

```
[Y91]  633:  mov ch,1
[H11]  634:  neg ax
[ULØ]  635: in1:
[W21]  636:  div si
[5G1]  637:  add dl,3Øh
[C42]  638:  mov [bx],dl
[FØ1]  639:  dec bx
[KØ1]  64Ø:  inc cl
[XF1]  641:  xor dl,dl
[571]  642:  or ax,ax
[6A1]  643:  jnz in1
[Z51]  644:  or ch,ch
[XWØ]  645:  jz in2
[VF3]  646:  mov byte ptr [bx],'-'
[FØ1]  647:  dec bx
[KØ1]  648:  inc cl
[YLØ]  649: in2:
[942]  65Ø:  mov [bx],cl
[8C1]  651:  push cs
[TC1]  652:  push bx
[L22]  653:  les di,[bp+8]
[EC1]  654:  push es
[3B1]  655:  push di
[5S1]  656:  call copys
[E31]  657:  pop ds
[131]  658:  pop bp
[7UØ]  659:  ret 2
[A12]  66Ø: intstr endp
[ØØØ]  661:
[EX2]  662: wordstr proc far
[TB1]  663:  push bp
[PG1]  664:  mov bp,sp
[452]  665:  mov ax,[bp+6]
[BC1]  666:  push ds
[8C1]  667:  push cs
[E31]  668:  pop ds
[RS3]  669:  mov bx,offset string+6
[TH1]  67Ø:  xor dx,dx
[1E1]  671:  mov si,1Ø
[HH1]  672:  mov cx,dx
[8WØ]  673: wor1:
[W21]  674:  div si
[5G1]  675:  add dl,3Øh
[C42]  676:  mov [bx],dl
[FØ1]  677:  dec bx
[KØ1]  678:  inc cl
[XF1]  679:  xor dl,dl
[571]  68Ø:  or ax,ax
[UG1]  681:  jnz wor1
[942]  682:  mov [bx],cl
[8C1]  683:  push cs
[TC1]  684:  push bx
[L22]  685:  les di,[bp+8]
[EC1]  686:  push es
[3B1]  687:  push di
[5S1]  688:  call copys
[E31]  689:  pop ds
[131]  69Ø:  pop bp
[7UØ]  691:  ret 2
[582]  692: wordstr endp
[ØØØ]  693:
[YT2]  694: scanleft proc far
[TB1]  695:  push bp
[PG1]  696:  mov bp,sp
[A52]  697:  mov dx,[bp+6]
[BC1]  698:  push ds
[T32]  699:  lds bx,[bp+8]
[D82]  7ØØ:  les di,[bp+12]
[M32]  7Ø1:  mov cl,[bx]
[6F1]  7Ø2:  xor ch,ch
[B61]  7Ø3:  or cl,cl
[VY1]  7Ø4:  jz scalleer
[FYØ]  7Ø5: scal1:
[M11]  7Ø6:  inc bx
[BØ1]  7Ø7:  inc di
[G32]  7Ø8:  mov al,[bx]
[ZD1]  7Ø9:  cmp al,dl
[Ø91]  71Ø:  jz scal2
[EH2]  711:  mov es:[di],al
[2P1]  712:  loop scal1
[852]  713:  mov bx,[bp+8]
[M32]  714:  mov cl,[bx]
[T92]  715:  mov di,[bp+12]
[KH2]  716:  mov es:[di],cl
[LYØ]  717: scal3:
[E31]  718:  pop ds
[131]  719:  pop bp
[FUØ]  72Ø:  ret 6
[HYØ]  721: scal2:
[LG1]  722:  mov cx,di
[T92]  723:  mov di,[bp+12]
[1F1]  724:  sub cx,di
[HØ1]  725:  dec cx
[ØG1]  726: scalleer:
[KH2]  727:  mov es:[di],cl
[ØG1]  728:  jmp scal3
[Z52]  729: scanleft endp
[ØØØ]  73Ø:
[P52]  731: copy proc far
[TB1]  732:  push bp
```

```
[PG1]  733:  mov bp,sp
[BC1]  734:  push ds
[A52]  735:  mov dx,[bp+6]
[A52]  736:  mov cx,[bp+8]
[F92]  737:  lds bx,[bp+1Ø]
[BZ3]  738:  mov di,offset string+255
[8C1]  739:  push cs
[G31]  74Ø:  pop es
[ØF1]  741:  xor ah,ah
[ØF1]  742:  mov dh,ah
[YE1]  743:  mov ch,ah
[3E1]  744:  cmp cl,dl
[C31]  745:  ja cop1
[422]  746:  cmp dl,[bx]
[C31]  747:  ja cop1
[B61]  748:  or cl,cl
[R51]  749:  jz cop1
[ED1]  75Ø:  add cx,bx
[ED1]  751:  add bx,dx
[TUØ]  752: cop2:
[G32]  753:  mov al,[bx]
[EH2]  754:  mov es:[di],al
[2Ø1]  755:  inc ah
[FØ1]  756:  dec bx
[6ZØ]  757:  dec di
[WF1]  758:  cmp bx,cx
[BC1]  759:  jnb cop2
[SUØ]  76Ø: cop1:
[864]  761:  mov byte ptr es:[di],ah
[8C1]  762:  push cs
[3B1]  763:  push di
[H82]  764:  les di,[bp+14]
[EC1]  765:  push es
[3B1]  766:  push di
[5S1]  767:  call copys
[E31]  768:  pop ds
[131]  769:  pop bp
[LUØ]  77Ø:  ret 8
[RH1]  771: copy endp
[ØØØ]  772:
[3Z2]  773: scanright proc far
[TB1]  774:  push bp
[PG1]  775:  mov bp,sp
[A52]  776:  mov dx,[bp+6]
[BC1]  777:  push ds
[T32]  778:  lds bx,[bp+8]
[M32]  779:  mov cl,[bx]
[AF1]  78Ø:  mov dh,cl
[6F1]  781:  xor ch,ch
[B61]  782:  or cl,cl
[791]  783:  jz scan1
[DD1]  784:  add bx,cx
[8C1]  785:  push cs
[G31]  786:  pop es
[PF3]  787:  mov di,offset string
[4C1]  788:  add di,cx
[LYØ]  789: scan2:
[G32]  79Ø:  mov al,[bx]
[ZD1]  791:  cmp al,dl
[791]  792:  jz scan1
[EH2]  793:  mov es:[di],al
[FØ1]  794:  dec bx
[6ZØ]  795:  dec di
[9P1]  796:  loop scan2
[G32]  797:  mov al,[bx]
[EH2]  798:  mov es:[di],al
[NYØ]  799: scan3:
[EC1]  8ØØ:  push es
[3B1]  8Ø1:  push di
[D82]  8Ø2:  les di,[bp+12]
[EC1]  8Ø3:  push es
[3B1]  8Ø4:  push di
[5S1]  8Ø5:  call copys
[E31]  8Ø6:  pop ds
[131]  8Ø7:  pop bp
[FUØ]  8Ø8:  ret 6
[HYØ]  8Ø9: scan1:
[SD1]  81Ø:  sub dh,cl
[8H2]  811:  mov es:[di],dh
[2G1]  812:  jmp scan3
[6C2]  813: scanright endp
[ØØØ]  814:
[X23]  815: vorkommen proc far
[TB1]  816:  push bp
[PG1]  817:  mov bp,sp
[A52]  818:  mov dx,[bp+6]
[BC1]  819:  push ds
[T32]  82Ø:  lds bx,[bp+8]
[PF3]  821:  mov di,offset string
[8C1]  822:  push cs
[G31]  823:  pop es
[9F1]  824:  xor dh,dh
[M32]  825:  mov cl,[bx]
[B61]  826:  or cl,cl
[S61]  827:  jz vor3
[1F1]  828:  mov ch,dh
[XE1]  829:  mov ah,dh
[BØ1]  83Ø:  inc di
[6WØ]  831: vor1:
[M11]  832:  inc bx
```

```
[2Ø1]  833:  inc ah
[G32]  834:  mov al,[bx]
[ZD1]  835:  cmp al,dl
[TG1]  836:  jnz vor2
[2H2]  837:  mov es:[di],ah
[8Ø1]  838:  inc dh
[BØ1]  839:  inc di
[7WØ]  84Ø: vor2:
[3K1]  841:  loop vor1
[PF3]  842:  mov di,offset string
[8WØ]  843: vor3:
[8H2]  844:  mov es:[di],dh
[L72]  845:  mov [bp+8],di
[RF2]  846:  mov [bp+1Ø],cs
[E31]  847:  pop ds
[131]  848:  pop bp
[7UØ]  849:  ret 2
[ØG2]  85Ø: vorkommen endp
[ØØØ]  851:
[TB3]  852: getposfeld proc far
[TB1]  853:  push bp
[PG1]  854:  mov bp,sp
[H22]  855:  les di,[bp+6]
[2B2]  856:  mov dx,[bp+1Ø]
[BC1]  857:  push ds
[L92]  858:  lds bx,[bp+12]
[9F1]  859:  xor dh,dh
[M32]  86Ø:  mov cl,[bx]
[B61]  861:  or cl,cl
[V51]  862:  jz get3
[1F1]  863:  mov ch,dh
[XE1]  864:  mov ah,dh
[BØ1]  865:  inc di
[KUØ]  866: get1:
[M11]  867:  inc bx
[2Ø1]  868:  inc ah
[G32]  869:  mov al,[bx]
[ZD1]  87Ø:  cmp al,dl
[5F1]  871:  jnz get2
[2H2]  872:  mov es:[di],ah
[8Ø1]  873:  inc dh
[BØ1]  874:  inc di
[LUØ]  875: get2:
[GF1]  876:  loop get1
[X32]  877:  mov di,[bp+6]
[MUØ]  878: get3:
[8H2]  879:  mov es:[di],dh
[E31]  88Ø:  pop ds
[131]  881:  pop bp
[NYØ]  882:  ret 1Ø
[2P2]  883: getposfeld endp
[ØØØ]  884:
[CP2]  885: alstyp proc far
[TB1]  886:  push bp
[PG1]  887:  mov bp,sp
[BC1]  888:  push ds
[H22]  889:  les di,[bp+6]
[F92]  89Ø:  lds bx,[bp+1Ø]
[M32]  891:  mov cl,[bx]
[6F1]  892:  xor ch,ch
[KH2]  893:  mov es:[di],cl
[RUØ]  894: als1:
[M11]  895:  inc bx
[BØ1]  896:  inc di
[G32]  897:  mov al,[bx]
[EH2]  898:  mov es:[di],al
[TF1]  899:  loop als1
[E31]  9ØØ:  pop ds
[131]  9Ø1:  pop bp
[LUØ]  9Ø2:  ret 8
[NØ2]  9Ø3: alstyp endp
[ØØØ]  9Ø4:
[TF1]  9Ø5: CODE ends
[4MØ]  9Ø6:  end
```

Listing 5. »strings.asm« beinhaltet die String-Behandlungsroutinen

```
[Y54]    1: ; Programm: windows.asm
[DY6]    2: ; Funktion: Fenster- und Menübehandlung
[5Y4]    3: ; Sprache:  MASM ab Version 4.Ø
[ZC4]    4: ; Autor:    Frank Emminghaus
[FU4]    5: ; (c)1991 DMV Widuch GmbH & Co. KG
[ØØØ]    6:
[H22]    7: CODE segment
[EA2]    8: assume cs:CODE
[Ø75]    9: public  offset,cursor,nocursor
[ØØØ]   1Ø:
[5M2]   11: offset proc far
[TB1]   12:  push bp
[PG1]   13:  mov bp,sp
[452]   14:  mov ax,[bp+6]
[ØF1]   15:  xor ah,ah
[C52]   16:  mov dx,[bp+8]
[9F1]   17:  xor dh,dh
[DØ1]   18:  dec ax
```

Listing 6. »windows.asm« kümmert sich um die Fenster- und Menübehandlung

```
[LØ1]   19:  dec dx
[GA1]   2Ø:  mov cl,7
[CH1]   21:  mov bx,ax
[3F1]   22:  shl ax,cl
[VR1]   23:  xchg ax,bx
[EA1]   24:  mov cl,5
[3F1]   25:  shl ax,cl
[AD1]   26:  add ax,bx
[Z91]   27:  shl dx,1
[CD1]   28:  add ax,dx
[131]   29:  pop bp
[BUØ]   3Ø:  ret 4
[EYl]   31: offset endp
[ØØØ]   32:
[SP2]   33: cursor proc far
[TW1]   34:  mov cx,Ø8Ø9h
[UV1]   35:  mov ax,1ØØh
[GZØ]   36:  int 16
[3PØ]   37:  ret
[112]   38: cursor endp
[ØØØ]   39:
[ZX2]   4Ø: nocursor proc far
[HV1]   41:  mov cx,3Ø3Øh
[UV1]   42:  mov ax,1ØØh
[GZØ]   43:  int 16
[3PØ]   44:  ret
[ØA2]   45: nocursor endp
[ØØØ]   46:
[TF1]   47: CODE ends
[4MØ]   48:  end
```

Listing 6. »windows.asm« kümmert sich um die Fenster- und Menübehandlung

```
[264]    1: { Programm: backwins.pas
[R75]    2:   Funktion: Oberflächen-Routine
[8Z5]    3:   Sprache:  Turbo Pascal ab Version 5.5
[R94]    4:   Autor:    Frank Emminghaus
[P35]    5:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    6:
[XD2]    7: unit backwins;
[ØØØ]    8:
[SK1]    9: interface
[R87]   1Ø: uses dosadd,windows,maus,dos,crt,strings;
[ØØØ]   11:
[ØØØ]   12:
[STØ]   13: TYPE
[MH4]   14:  wrec=record lo,hi:byte end;
[ZY4]   15:  longrec=record lo,hi:word end;
[462]   16:  lwahl=record
[8K2]   17:          size,zlw:byte;
[A15]   18:          sflag: array[1..8Ø] of boolean;
[X85]   19:          sub: array[1..8Ø] of string[1ØØ]
[ZNØ]   2Ø:          end;
[LS5]   21:      storerec=record name:string[12];
[9A2]   22:                      time:longint;
[VB2]   23:                      size:longint;
[WM1]   24:                      attr:byte
[ZNØ]   25:                      end;
[ØØØ]   26:
[M82]   27:       lpoi = ^lwahl;
[DF5]   28:       timerec=record l:char; a:byte end;
[HY4]   29:       timear=array[1..5] of timerec;
[9B5]   3Ø:       ptrec=record ofs, seg: word end;
[C14]   31:       direintrag= string[8];
[8T5]   32:       sektors=record sekt,ofs:word end;
[2M5]   33:       puffermust=array[Ø..1Ø23] of byte;
[YC4]   34:       recogn=array[Ø..5] of byte;
[ØØØ]   35:
[ØØØ]   36:
[RNØ]   37: VAR
[666]   38:   dateifeld:  array[1..5ØØ] of storerec;
[DX5]   39:   sektfeld:   array[1..3ØØ] of sektors;
[6N4]   4Ø:   rec:        array[Ø..4] of byte;
[1Z4]   41:   endrec:     array[Ø..5] of byte;
[TC6]   42:   subdrs:     array[1..25Ø] of string[14];
[WT5]   43:   selected:   array[Ø..25Ø] of boolean;
[L25]   44:   lwfeld:     array[1..2Ø] of lpoi;
[914]   45:   puffer:     ^puffermust;
[9X2]   46:   time:       ^timear;
[SH3]   47:   lwstring:  string[5];
[FG3]   48:   namstr:    string[12];
[6U4]   49:   path,s,p1,p2,attstr,aktpath,
[8R4]   5Ø:   orgpath:                    string;
[1D5]   51:   pathsize,i,diskno,dirno,qdatei,
[RE6]   52:   zdatei,ondisk,zuschreiben,nextsektor,
[PD4]   53:   dfsize,anf:                 word;
[CP5]   54:   fehl,nob,indisk,abbruch,teilrest,
[D27]   55:   wpath,sflag,ende,werror,error,on,verify,
[KX4]   56:   restflag,timeflag,zwingflag,
[VZ4]   57:   ersatz,least:               boolean;
[UØ5]   58:   searchtime,nochfrei,freesize,
[NK4]   59:   aktfree:                    longint;
[C76]   6Ø:   col1,col2,col3,col4,col5,col6,col7,col8,
[516]   61:   hot1,hot2,hot3,sel1,sel2,lw,tic,tic1,
[SD6]   62:   kor,kor1,secs,rlw,orglw,dup:      byte;
[4Y1]   63:   a: menufeld;
[242]   64:   r: registers;
```

```
[GC1]   65:   ch: char;
[UF2]   66:   r1,r2: storerec;
[9Ø3]   67:   sektpoi: pointer;
[H45]   68:   freebyte,freesektor: integer;
[5Z2]   69:   men: submenufeld;
[EC4]   7Ø:   oldbreak, oldint: pointer;
[ØØØ]   71:
[ØØØ]   72:
[334]   73: procedure backohandler;
[274]   74: procedure restohandler;
[ES2]   75: procedure quit;
[XG3]   76: procedure nobutton;
[DY2]   77: procedure clrbuf;
[3Ø5]   78: procedure infoline(s:string);
[9F3]   79: procedure timeinit;
[EN2]   8Ø: procedure beep;
[7V6]   81: procedure writefiles(orand,urand:byte);
[TD6]   82: procedure writes(lw,start,zahl:word;
[ZZ4]   83:           p:pointer; var error:boolean);
[G76]   84: procedure reads(lw,start,zahl:word;
[ZZ4]   85:           p:pointer; var error:boolean);
[7Y6]   86: procedure changedir(p:string; var error:
[XG1]   87:                      boolean);
[ØØØ]   88:
[4L2]   89: implementation
[6A2]   9Ø: {$l neuint}
[ØØØ]   91:
[RNØ]   92: VAR
[AB3]   93:   b:menufeld; q:word;
[FT5]   94:   ext:      array[1..25Ø] of string[12];
[P75]   95:   wpos,vor: array[Ø..25Ø] of byte;
[CP4]   96:   trans:    array[1..2Ø] of byte;
[6F6]   97:   last,ba,unter: array[Ø..25Ø] of boolean;
[ZE3]   98:   subdrsize,qlw :byte;
[HD2]   99:   lwstr: string;
[Z83]  1ØØ:   datei: file of lwahl;
[ØØØ]  1Ø1:
[LZ4]  1Ø2: procedure getquell; forward;
[SR5]  1Ø3: procedure getrestquell; forward;
[ØØØ]  1Ø4:
[ZWØ]  1Ø5: {$f+}
[AT4]  1Ø6: procedure neuint; external;
[1XØ]  1Ø7: {$f-}
[ØØØ]  1Ø8:
[EN2]  1Ø9:  procedure beep;
[MUØ]  11Ø:  begin
[1Y4]  111:  sound(8ØØ); delay(1ØØ); nosound
[ZNØ]  112:  end;
[ØØØ]  113:
[XG3]  114:  procedure nobutton;
[MUØ]  115:  begin
[ZW6]  116:  repeat until not button1 and not button2
[ZNØ]  117:  end;
[ØØØ]  118:
[DY2]  119:  procedure clrbuf;
[6Y2]  12Ø:  var r:registers;
[MUØ]  121:  begin
[5Ø3]  122:  r.ax:=$cØØ; msdos(r)
[ZNØ]  123:  end;
[ØØØ]  124:
[9F3]  125:  procedure timeinit;
[274]  126:  var std,min,sec,hs:word;
[MUØ]  127:  begin
[W23]  128:  time:=ptr(bild,148);
[UE2]  129:  tic:=18; kor:=5;
[LL5]  13Ø:  for i:=1 to 5 do time^[i].a:=col4;
[244]  131:  gettime(std,min,sec,hs);
[EL3]  132:  getintvec($1c,oldint);
[1V4]  133:  time^[5].l:=chr(48+min mod 1Ø);
[YV4]  134:  time^[4].l:=chr(48+min div 1Ø);
[AH2]  135:  time^[3].l:=':';
[LU4]  136:  time^[2].l:=chr(48+std mod 1Ø);
[GV4]  137:  time^[1].l:=chr(48+std div 1Ø);
[7P1]  138:  secs:=sec;
[4V3]  139:  setintvec($1c,@neuint);
[ZNØ]  14Ø:  end;
[ØØØ]  141:
[7Y6]  142:  procedure changedir(p:string; var error:
[XG1]  143:                      boolean);
[9X5]  144:  var  lauf:string[3]; r:string[1Ø];
[PD5]  145:       j,b:string; esc:boolean; a:byte;
[1Ø2]  146:       io:boolean;
[Z41]  147:       label 1;
[ØØØ]  148:
[5C6]  149:    procedure exnicht(var error:boolean);
[VY1]  15Ø:    var i:byte;
[MUØ]  151:    begin
[Y24]  152:    a:=textattr; on:=false;
[1R5]  153:    textattr:=col2; normrahmen:=col2;
[GW3]  154:    rahmen(5,1Ø,7Ø,5,'Fehler');
[DC1]  155:    nomaus;
[P31]  156:    repeat
[5Y4]  157:    textattr:=col2; clrscr; beep;
[725]  158:    writeln(' Verzeichnis nicht '+
[BZ1]  159:            'gefunden:');
[6B2]  16Ø:    writeln(' '+p);
[HV5]  161:    writeln(' Bitte neues Verzeichnis'
[772]  162:          +' eingeben:');
[K64]  163:    writeln; textattr:=col6;
[ZY2]  164:    gotoxy(2,wherey);
```

```
[VU4]  165:    b:='<RET> für Verz: '+aktpath;
[8Y1]  166:    infoline(b);
[C84]  167:    for i:=1 to 65 do write(' ');
[ZY2]  168:    gotoxy(2,wherey);
[4S2]  169:    eingabe(j,esc,68);
[UP3]  170:    if j='' then j:=aktpath;
[KV2]  171:    zwingflag:=true;
[S73]  172:    changedir(j,error);
[ZY2]  173:    zwingflag:=false;
[7Y3]  174:    until not error or esc;
[6Y3]  175:    if esc then error:=true;
[BM1]  176:    nocursor;
[GH2]  177:    restorescreen;
[LY1]  178:    infoline('');
[7D3]  179:    showmaus; on:=true;
[DU4]  180:    textattr:=a; normrahmen:=a;
[4U3]  181:    if not error then begin
[9X0]  182:                        p1:=p;
[FR0]  183:                        p2:=j
[4M0]  184:                        end
[ZN0]  185:    end;
[000]  186:
[000]  187:
[MU0]  188:  begin
[M35]  189:  j:=p; esc:=false; error:=false;
[9L3]  190:  lauf:=scanleft(j,'\');
[H54]  191:  delleft(j,length(lauf)+1);
[NE4]  192:  {$i-} chdir(lauf+'\'); {$i+}
[T53]  193:  if ioresult<>0 then
[MU0]  194:    begin
[5Y5]  195:    if not zwingflag then exnicht(error)
[E03]  196:              else error:=true;
[DZ0]  197:    goto 1
[ZN0]  198:    end;
[TZ2]  199:  while j<>'' do begin
[ZZ2]  200:       r:=scanleft(j,'\');
[DX0]  201:       {$i-}
[UD1]  202:       chdir(r);
[BX0]  203:       {$i+}
[703]  204:       io:=(ioresult<>0);
[VV3]  205:       if io and zwingflag then
[MU0]  206:          begin
[V93]  207:          {$i-} mkdir(r) {$i+};
[T53]  208:          if ioresult<>0 then
[MU0]  209:             begin
[A83]  210:             error:=true; goto 1
[ZN0]  211:             end;
[4A1]  212:          chdir(r)
[ZN0]  213:          end;
[AG4]  214:       if io and not zwingflag then
[MU0]  215:          begin
[BS2]  216:          exnicht(error);
[DZ0]  217:          goto 1
[ZN0]  218:          end;
[FH3]  219:       delleft(j,length(r)+1)
[ZN0]  220:       end;
[H12]  221:  aktpath:=p;
[G80]  222:  1:
[ZN0]  223:  end;
[000]  224:
[CC5]  225:  procedure neubreak; interrupt;
[6Y2]  226:  var r:registers;
[MU0]  227:  begin
[4L1]  228:  r.ax:=$c00;
[0S1]  229:  intr($21,r);
[6M3]  230:  setintvec($1c,oldint);
[FX3]  231:  setintvec($1b,oldbreak);
[051]  232:  r.ah:=0;
[G51]  233:  r.dl:=0;
[YR1]  234:  intr($13,r);
[0D1]  235:  cursor;
[XY3]  236:  nomaus; window(1,1,80,25);
[Z25]  237:  textattr:=lightgray;clrscr;
[UR2]  238:  textattr:=col7;
[MT5]  239:  write('     SBackup (c) Copyright 1991'
[914]  240:       +' DMV Widuch GmbH & Co. KG'
[KX3]  241:       +' DOS International    ');
[KR3]  242:  textattr:=lightgray;
[CV3]  243:  changedir(orgpath,nob);
[RP0]  244:  halt
[ZN0]  245:  end;
[000]  246:
[305]  247:  procedure infoline(s:string);
[YA3]  248:  var i:byte; w:word;
[MU0]  249:   begin
[DC1]  250:   nomaus;
[8U3]  251:   for i:=1 to length(s) do
[MU0]  252:      begin
[LP3]  253:      wrec(w).lo:=ord(s[i]);
[0S2]  254:      wrec(w).hi:=col5;
[4D4]  255:      memw[bild:offset(i,25)]:=w
[ZN0]  256:      end;
[N64]  257:   for i:=length(s)+1 to 80 do
[MU0]  258:      begin
[MD2]  259:      wrec(w).lo:=32;
[0S2]  260:      wrec(w).hi:=col5;
[4D4]  261:      memw[bild:offset(i,25)]:=w
[ZN0]  262:      end;
[NG1]  263:   showmaus
[ZN0]  264:   end;
```

```
[000]  265:
[ES2]  266:  procedure quit;
[6X4]  267:  var a:menufeld; t,h,q,s:byte;
[F72]  268:      r:registers;
[MU0]  269:   begin
[H64]  270:   q:=menumode;  h:=hotattr;
[HX4]  271:   hotattr:=hot1;  t:=textattr;
[584]  272:   a[2]:='^nein'; a[1]:='^ja';
[5R5]  273:   textattr:=col3; normrahmen:=col3;
[1W2]  274:   rahmen(30,12,20,4,'');
[DC1]  275:   nomaus;
[DB1]  276:   clrscr;
[NT6]  277:   s:=selectattribut; selectattribut:=sel2;
[6M4]  278:   writeln(' Sind Sie Sicher ?');
[BM1]  279:   showmaus;
[ST2]  280:   window(33,13,47,15);
[SZ1]  281:   menumode:=0;
[AL3]  282:   horizontalmenu(3,2,a);
[613]  283:   if not escflag then
[FY2]  284:      if menumode=1 then
[MU0]  285:         begin
[0D1]  286:         cursor;
[DC1]  287:         nomaus;
[GH2]  288:         restorescreen;
[GH2]  289:         window(1,1,80,25);
[KR3]  290:         textattr:=lightgray;
[DB1]  291:         clrscr;
[UR2]  292:         textattr:=col7;
[456]  293:   writeln('     SBackup (c) Copyright 1991'
[914]  294:        +' DMV Widuch GmbH & Co. KG'
[KX3]  295:        +' DOS International    ');
[KR3]  296:         textattr:=lightgray;
[4L1]  297:         r.ax:=$c00;
[0S1]  298:         intr($21,r);
[6M3]  299:         setintvec($1c,oldint);
[FX3]  300:         setintvec($1b,oldbreak);
[CV3]  301:         changedir(orgpath,nob);
[RP0]  302:         halt
[ZN0]  303:         end;
[XP5]  304:   textattr:=col1; normrahmen:=col1;
[A73]  305:   selectattribut:=s;
[LC4]  306:   hotattr:=h; textattr:=t;
[GH2]  307:   restorescreen;
[122]  308:   menumode:=q
[ZN0]  309:   end;
[000]  310:
[000]  311:
[VV5]  312:  procedure getsubdirs(path:string;
[2T4]  313:            vorg:byte; lastein:boolean);
[046]  314:  var sub: array[1..50] of string[12];
[XC3]  315:      subsize,i,pos:byte;
[000]  316:
[ZX5]  317:   function infeld(s:string):boolean;
[4Y6]  318:   var schl:searchrec; attr:word; p:string;
[MU0]  319:   begin
[BN1]  320:   p:=s+'\*.*';
[4Z1]  321:   subsize:=0;
[PG3]  322:   findfirst(p,$10,schl);
[EA3]  323:   while doserror=0 do
[MU0]  324:      begin
[A93]  325:      if schl.attr=$10 then
[MU0]  326:         begin
[T73]  327:         if (schl.name<>'.')and
[4A3]  328:            (schl.name<>'..') then
[MU0]  329:                begin
[382]  330:                inc(subsize);
[A64]  331:                sub[subsize]:=schl.name
[4M0]  332:                end
[ZN0]  333:         end;
[KD2]  334:         findnext(schl)
[ZN0]  335:     end;
[DM5]  336:   if doserror<>18 then infeld:=false
[DZ2]  337:                  else infeld:=true
[ZN0]  338:   end;
[000]  339:
[MU0]  340:  begin
[253]  341:  if infeld(path) then
[MU0]  342:    begin
[DN2]  343:    inc(subdrsize);
[YW2]  344:    pos:=subdrsize;
[CX3]  345:    vor[subdrsize]:=vorg;
[SM4]  346:    last[subdrsize]:=lastein;
[FL3]  347:    wpos[subdrsize]:=1+
[N44]  348:     wpos[vor[subdrsize]];
[3S6]  349:    subdrs[subdrsize]:=scanright(path,'\');
[VP0]  350:    i:=1;
[943]  351:    while i<subsize do
[MU0]  352:      begin
[8A3]  353:      getsubdirs(path+'\'+
[S13]  354:                 sub[i],pos,false);
[XX0]  355:      inc(i)
[ZN0]  356:      end;
[3P5]  357:    if subsize>0 then getsubdirs(path+
[FG4]  358:                 '\'+sub[subsize],pos,true)
[4M0]  359:    end
[ZN0]  360:  end;
[000]  361:
[0X4]  362:  procedure getdatum; forward;
```

Listing 7. »backwins.pas« stellt grundlegende Oberflächenroutinen

```
[RS4]  363:  procedure getziel; forward;
[C55]  364:  procedure saveconfig; forward;
[H35]  365:  procedure loadconfig; forward;
[ØØØ]  366:
[334]  367:  procedure backohandler;
[A84]  368:  var a:menufeld; c,q:byte;
[MUØ]  369:  begin
[Z22]  37Ø:  q:=menumode;
[SZ1]  371:  menumode:=Ø;
[P31]  372:  repeat
[2W1]  373:  if timeflag
[9Ø5]  374:     then a[1]:='^Uhrzeit/Datum √'
[WZ4]  375:     else a[1]:='^Uhrzeit/Datum  ';
[9Y5]  376:  if verify then a[2]:='^Verify √'
[GY3]  377:           else a[2]:='^Verify  ';
[6R3]  378:  a[3]:='^Quellaufwerk';
[SP3]  379:  a[4]:='^Ziellaufwerk';
[MU3]  38Ø:  a[5]:='Konf. ^sichern';
[ZC3]  381:  a[6]:='Konf. ^laden';
[1M3]  382:  verticalmenu(28,3,6,a);
[322]  383:  c:=menumode;
[613]  384:  if not escflag then
[MUØ]  385:   begin
[ØR2]  386:   case menumode of
[351]  387:   1: begin
[UØ4]  388:      timeflag:=not timeflag;
[B44]  389:      if timeflag then getdatum
[ZNØ]  39Ø:      end;
[HH3]  391:   2: verify:=not verify;
[LY1]  392:   3: getquell;
[EV1]  393:   4: getziel;
[E72]  394:   5: saveconfig;
[262]  395:   6: loadconfig;
[4MØ]  396:   end
[ZNØ]  397:   end;
[9Ø2]  398:  menumode:=c
[2N3]  399:  until escflag or (c>4);
[GH2]  4ØØ:  restorescreen;
[122]  4Ø1:  menumode:=q
[ZNØ]  4Ø2:  end;
[ØØØ]  4Ø3:
[274]  4Ø4:  procedure restohandler;
[WZ3]  4Ø5:  var a:menufeld; q:byte;
[MUØ]  4Ø6:  begin
[Z22]  4Ø7:  q:=menumode;
[SZ1]  4Ø8:  menumode:=Ø;
[P31]  4Ø9:  repeat
[1Ø2]  41Ø:  if zwingflag
[GH6]  411:    then a[1]:='Verzeichnis ^erzwingen √'
[3H6]  412:    else a[1]:='Verzeichnis ^erzwingen  ';
[9Y5]  413:  if verify then a[2]:='^Verify √'
[GY3]  414:           else a[2]:='^Verify  ';
[6R3]  415:  a[3]:='^Quellaufwerk';
[VL3]  416:  verticalmenu(48,3,3,a);
[613]  417:  if not escflag then
[ØR2]  418:     case menumode of
[PN4]  419:     1: zwingflag:=not zwingflag;
[HH3]  42Ø:     2: verify:=not verify;
[7P2]  421:     3: getrestquell
[ZNØ]  422:     end;
[1X4]  423:  until escflag or (menumode=3);
[GH2]  424:  restorescreen;
[W32]  425:  menumode:=q;
[ZNØ]  426:  end;
[ØØØ]  427:
[X56]  428:  procedure writepath(anf,ende:byte);
[9K2]  429:  var i,j,k:byte;
[MUØ]  43Ø:  begin
[VPØ]  431:  i:=1;
[1K3]  432:  while i<=subdrsize do
[MUØ]  433:    begin
[A52]  434:    if i>=anf then
[ØC2]  435:      if i<=ende then
[MUØ]  436:        begin
[P22]  437:        j:=wpos[i];
[ZPØ]  438:        k:=1;
[9V1]  439:        while j>2 do
[MUØ]  44Ø:          begin
[AA4]  441:          if ba[k] then write(' |')
[EF2]  442:                   else write('  ');
[D51]  443:          inc(k);
[UWØ]  444:          dec(j)
[ZNØ]  445:          end;
[8N1]  446:        if j>1 then
[NZ4]  447:           if last[i] then write(' └')
[PP3]  448:                      else write(' ├');
[E23]  449:        write(subdrs[i]);
[LK3]  45Ø:        if i<ende then writeln
[ZNØ]  451:        end;
[RW4]  452:    ba[wpos[i]-1]:=not last[i];
[XXØ]  453:    inc(i)
[4MØ]  454:    end
[ZNØ]  455:  end;
[ØØØ]  456:
[ØV4]  457:  procedure asksubdrs(i:byte);
[6E6]  458:  var a:menufeld; s:string; esc:boolean;
[CH1]  459:      h,q:byte;
[MUØ]  46Ø:  begin
[XP5]  461:  normrahmen:=col2; textattr:=col2;
[3K4]  462:  h:=hotattr; hotattr:=hot3;
```

```
[HT6]  463:  q:=selectattribut; selectattribut:=sel2;
[SR2]  464:  rahmen(3Ø,8,24,5,'');
[DB1]  465:  clrscr;
[614]  466:  writeln(' Backup aller');
[TM5]  467:  writeln(' Unterverzeichnisse ?');
[B94]  468:  a[1]:='^ja'; a[2]:='^nein';
[SZ1]  469:  menumode:=Ø;
[ZS2]  47Ø:  window(34,9,5Ø,13);
[BM1]  471:  showmaus;
[BL3]  472:  horizontalmenu(4,2,a);
[YL5]  473:  if menumode=1 then unter[i]:=true
[ES3]  474:                else unter[i]:=false;
[ØT2]  475:  window(31,9,53,12);
[DC1]  476:  nomaus;
[DB1]  477:  clrscr;
[FC6]  478:  infoline('   Die Suchmasken sind äqui'
[TK5]  479:       +'valent zu denen in DOS. Voreins'
[9V2]  48Ø:       +'tellung ist *.*');
[CM4]  481:  writeln(' Bitte Suchmaske');
[BB3]  482:  writeln(' eingeben');
[RR2]  483:  textattr:=col6;
[5G4]  484:  gotoxy(4,4); write('
[WV1]  485:  gotoxy(4,4);
[WR2]  486:  eingabe(s,esc,12);
[BM1]  487:  nocursor;
[BB2]  488:  if not esc then
[MU3]  489:     if s='' then ext[i]:='*.*'
[ML2]  49Ø:             else ext[i]:=s
[V33]  491:   else ext[i]:='*.*';
[GH2]  492:  restorescreen;
[VL6]  493:  infoline(' Bitte Pfade auswählen, Pfeil'
[C75]  494:       + 'zeichen anklicken, um mit Maus'
[5P2]  495:       + ' zu scrollen');
[XP5]  496:  textattr:=col1; normrahmen:=col1;
[765]  497:  hotattr:=h; selectattribut:=q
[ZNØ]  498:  end;
[ØØØ]  499:
[TY2]  5ØØ:  procedure pruefe;
[FL5]  5Ø1:  var q:string[1ØØ]; ex: string[12];
[Y62]  5Ø2:      s:string[14];
[YD5]  5Ø3:      aktpos,i,b,j:word; found:boolean;
[SX1]  5Ø4:      label raus;
[MUØ]  5Ø5:  begin
[AW3]  5Ø6:  b:=lwfeld[qlw]^.size;
[1H2]  5Ø7:  for i:=1 to b do
[MUØ]  5Ø8:    begin
[1D4]  5Ø9:    q:=lwfeld[qlw]^.sub[i];
[8C3]  51Ø:    ex:=scanright(q,'\');
[S12]  511:    delleft(q,3);
[4S3]  512:    delright(q,length(ex));
[KP1]  513:    aktpos:=1;
[2P1]  514:    if q='' then
[MUØ]  515:       begin
[KP1]  516:       aktpos:=1;
[983]  517:       selected[1]:=true;
[T22]  518:       ext[1]:=ex;
[AW5]  519:       unter[1]:=lwfeld[qlw]^.sflag[i];
[FK1]  52Ø:       goto raus
[ZNØ]  521:       end;
[242]  522:    while q<>'' do
[MUØ]  523:      begin
[FØ3]  524:      s:=scanleft(q,'\');
[V74]  525:      j:=aktpos+1; found:=false;
[8Ø6]  526:      while (j<=subdrsize) and not found do
[MUØ]  527:        begin
[384]  528:        found:=(s=subdrs[j])and
[YW2]  529:                    (vor[j]=aktpos);
[ZXØ]  53Ø:        inc(j)
[ZNØ]  531:        end;
[641]  532:      dec(j);
[2H4]  533:      if not found then goto raus;
[4S1]  534:      aktpos:=j;
[ØK3]  535:      delleft(q,length(s)+1)
[ZNØ]  536:      end;
[594]  537:    selected[aktpos]:=true;
[933]  538:    ext[aktpos]:=ex;
[FZ6]  539:    unter[aktpos]:=lwfeld[qlw]^.sflag[i];
[2WØ]  54Ø:    raus:
[9R6]  541:    if unter[aktpos] then subdrs[aktpos]:=
[9W3]  542:                         subdrs[aktpos]+' δ'
[NX3]  543:                       else subdrs[aktpos]:=
[MZ3]  544:                         subdrs[aktpos]+' √';
[4MØ]  545:    end
[ZNØ]  546:  end;
[ØØØ]  547:
[RE3]  548:  procedure markdirs;
[H66]  549:  var k,i,attr,curspos,ch,ypos,orand,
[BT4]  55Ø:      urand:byte; fertig: boolean;
[YL1]  551:      w: string;
[ØØØ]  552:
[TZ2]  553:      procedure invers;
[MUØ]  554:      begin
[EV2]  555:      attr:=textattr;
[4L4]  556:      textattr:=selectattribut;
[3B5]  557:      if curspos=1 then gotoxy(1,ypos)
[VD2]  558:         else gotoxy(2+
[C15]  559:           (wpos[curspos]-1) shl 1,ypos);
[N84]  56Ø:      write(subdrs[curspos]);
[CL2]  561:      textattr:=attr
[ZNØ]  562:      end;
```

```
[ØØØ]  563:
[5H3]  564:       procedure writeups;
[BY1]  565:       var a:byte;
[MUØ]  566:       begin
[262]  567:       a:=textattr;
[RR2]  568:       textattr:=col6;
[LF4]  569:       gotoxy(3,16); write(chr(24));
[UF4]  57Ø:       gotoxy(7,16); write(chr(25));
[KH4]  571:       textattr:=a; gotoxy(1Ø,16);
[WF5]  572:       write('wenn fertig, <ESC> (rechte '
[2U3]  573:             + 'Maustaste) drücken');
[NT1]  574:       gotoxy(1,1)
[ZNØ]  575:       end;
[ØØØ]  576:
[MUØ]  577:    begin
[RØ2]  578:    wpos[Ø]:=Ø;
[DC1]  579:    nomaus;
[Y62]  58Ø:    subdrsize:=Ø;
[6U5]  581:    textattr:=col4; normrahmen:= col4;
[CS2]  582:    rahmen(25,9,27,3,'');
[DB1]  583:    clrscr;
[XF1]  584:    writeln;
[385]  585:    writeln('  Lese Verzeichnis...');
[BM1]  586:    showmaus;
[LB4]  587:    getsubdirs(lwstr,Ø,false);
[GH2]  588:    restorescreen;
[XP5]  589:    textattr:=col1; normrahmen:=col1;
[L14]  59Ø:    for i:=1 to subdrsize do
[ØG3]  591:                  selected[i]:=false;
[WA4]  592:    subdrs[1]:=subdrs[1]+'\';
[DK5]  593:    if lwfeld[qlw]<>nil then pruefe;
[DC1]  594:    nomaus;
[KC5]  595:    rahmen(11,5,55,16,'Verzeichnisse');
[N63]  596:    clrscr;  writeups;
[US2]  597:    window(12,6,56,2Ø);
[KC5]  598:    fertig:=false;  ch:=Ø; curspos:=1;
[HX2]  599:    ypos:=1; orand:=1;
[7L5]  6ØØ:    urand:=15; writepath(orand,urand);
[2D1]  6Ø1:    invers;
[VL6]  6Ø2:    infoline(' Bitte Pfade auswählen, Pfeil'
[C75]  6Ø3:        + 'zeichen anklicken, um mit Maus'
[5P2]  6Ø4:        + ' zu scrollen');
[BM1]  6Ø5:    showmaus;
[Z83]  6Ø6:    while not fertig do
[MUØ]  6Ø7:      begin
[ØT5]  6Ø8:      if keypressed then ch:=ord(readkey);
[YG3]  6Ø9:      if (ch=72)or(ch=8Ø) then
[MUØ]  61Ø:         begin
[DC1]  611:         nomaus;
[3B5]  612:         if curspos=1 then gotoxy(1,ypos)
[DE5]  613:            else gotoxy(2+(wpos[curspos]-1)
[HØ2]  614:                    shl 1 ,ypos);
[N84]  615:         write(subdrs[curspos]);
[VX1]  616:         if ch=8Ø then
[MUØ]  617:            begin
[UB4]  618:            if curspos<subdrsize then
[8A2]  619:                     inc(curspos);
[7L3]  62Ø:            if curspos>urand then
[MUØ]  621:               begin
[DB1]  622:               clrscr;
[5W1]  623:               inc(orand);
[HW1]  624:               inc(urand);
[1T3]  625:               writepath(orand,urand)
[4MØ]  626:               end
[ZNØ]  627:           end;
[TX1]  628:       if ch=72 then
[MUØ]  629:          begin
[D55]  63Ø:          if curspos>1 then dec(curspos);
[TK3]  631:          if curspos<orand then
[MUØ]  632:               begin
[DB1]  633:               clrscr;
[ØV1]  634:               dec(orand);
[CV1]  635:               dec(urand);
[1T3]  636:               writepath(orand,urand)
[4MØ]  637:               end
[ZNØ]  638:           end;
[YU3]  639:       ypos:=curspos-orand+1;
[2D1]  64Ø:       invers;
[BM1]  641:       showmaus;
[3RØ]  642:       ch:=Ø
[ZNØ]  643:       end;
[HZ3]  644:    if (ch=27)or button2 then
[MUØ]  645:       begin
[G92]  646:       fertig:=true;
[U54]  647:       repeat until not button2
[ZNØ]  648:       end;
[5G2]  649:    if button1 then
[D92]  65Ø:     if akty>4 then
[992]  651:      if aktx>4 then
[ØE2]  652:       if aktx<62 then
[PD2]  653:        if akty<2Ø then
[MUØ]  654:           begin
[H34]  655:           if curspos+akty-4-ypos<=
[2G2]  656:              subdrsize then
[MUØ]  657:               begin
[DC1]  658:               nomaus;
[WU2]  659:               if curspos=1 then
[393]  66Ø:                   gotoxy(1,ypos) else
[ED4]  661:                   gotoxy(2+(wpos[curspos]
[FC2]  662:                         -1) shl 1,ypos);
```

```
[N84]  663:               write(subdrs[curspos]);
[DN1]  664:               curspos:=
[7K3]  665:                  curspos+akty-4-ypos;
[W52]  666:               ypos:=akty-4;
[2D1]  667:               invers;
[A31]  668:               ch:=13;
[BM1]  669:               showmaus;
[T54]  67Ø:               repeat until not button1
[4MØ]  671:               end
[4MØ]  672:          end
[G93]  673:           else if akty=2Ø then
[UD2]  674:                if aktx=13 then
[MUØ]  675:                   begin
[GC2]  676:                   if orand>1 then
[MUØ]  677:                      begin
[DC1]  678:                      nomaus;
[DB1]  679:                      clrscr;
[ØV1]  68Ø:                      dec(orand);
[CV1]  681:                      dec(urand);
[392]  682:                      dec(curspos);
[CT2]  683:                      writepath(orand,
[171]  684:                                urand);
[2D1]  685:                      invers;
[BM1]  686:                      showmaus;
[1E1]  687:                      delay(1ØØ)
[4MØ]  688:                      end
[4MØ]  689:                   end
[ZRØ]  69Ø:                   else
[6E2]  691:                    if aktx=17 then
[MUØ]  692:                       begin
[W81]  693:                       if urand
[8S2]  694:                        <subdrsize then
[MUØ]  695:                         begin
[DC1]  696:                         nomaus;
[DB1]  697:                         clrscr;
[5W1]  698:                         inc(orand);
[HW1]  699:                         inc(urand);
[8A2]  7ØØ:                         inc(curspos);
[YL1]  7Ø1:                         writepath
[T82]  7Ø2:                         (orand,urand);
[2D1]  7Ø3:                         invers;
[BM1]  7Ø4:                         showmaus;
[1E1]  7Ø5:                         delay(1ØØ)
[4MØ]  7Ø6:                         end
[ZNØ]  7Ø7:                       end;
[2X1]  7Ø8:    if ch=13 then
[MUØ]  7Ø9:       begin
[DC1]  71Ø:       nomaus;
[V35]  711:       if not selected [curspos] then
[4C3]  712:           asksubdrs(curspos);
[3B5]  713:       if curspos=1 then gotoxy(1,ypos)
[ZRØ]  714:         else
[GD5]  715:         gotoxy(2+(wpos[curspos]-1) shl 1
[Z71]  716:           ,ypos);
[N84]  717:       write(subdrs[curspos]);
[AL3]  718:       selected[curspos]:=
[K34]  719:          not selected[curspos];
[HB4]  72Ø:       if selected[curspos] then
[7X3]  721:        if unter[curspos] then
[S93]  722:           subdrs[curspos]:=
[G94]  723:                  subdrs[curspos]+' δ'
[M34]  724:         else subdrs[curspos]:=
[GB4]  725:                  subdrs[curspos]+' √'
[ZRØ]  726:         else
[VY4]  727:           delright(subdrs[curspos],2);
[DB1]  728:       clrscr;
[DØ4]  729:       writepath(orand,urand);
[2D1]  73Ø:       invers;
[BM1]  731:       showmaus;
[3RØ]  732:       ch:=Ø
[4MØ]  733:       end
[ZNØ]  734:    end;
[GH2]  735:  restorescreen;
[LY1]  736:  infoline('');
[L14]  737:  for i:=1 to subdrsize do
[K73]  738:    if selected[i] then
[6U3]  739:            delright(subdrs[i],2);
[NS3]  74Ø:  delright(subdrs[1],1);
[114]  741:  if lwfeld[qlw]=nil then
[U53]  742:                new(lwfeld[qlw]);
[FT3]  743:  lwfeld[qlw]^.size:=Ø;
[3X3]  744:  lwfeld[qlw]^.zlw:=lw;
[L14]  745:  for i:=1 to subdrsize do
[MUØ]  746:       begin
[K73]  747:       if selected[i] then
[MUØ]  748:         begin
[894]  749:         w:=subdrs[i]+'\'+ext[i];
[2U1]  75Ø:         k:=vor[i];
[6U2]  751:         while k>Ø do begin
[YA3]  752:                     w:=subdrs[k]+'\'+w;
[LM1]  753:                     k:=vor[k]
[ZNØ]  754:                     end;
[Y74]  755:         inc(lwfeld[qlw]^.size);
[VØ4]  756:         ch:=lwfeld[qlw]^.size;
[LH4]  757:         lwfeld[qlw]^.sub[ch]:=w;
[M36]  758:         lwfeld[qlw]^.sflag[ch]:=unter[i]
[4MØ]  759:         end
[ZNØ]  76Ø:       end;
```

Listing 7. »backwins.pas« stellt grundlegende Oberflächenroutinen

```
[7U4]  761:   if lwfeld[qlw]^.size=Ø then
[MUØ]  762:      begin
[XW3]  763:      dispose(lwfeld[qlw]);
[U13]  764:      lwfeld[qlw]:=nil
[4MØ]  765:      end
[ZNØ]  766:   end;
[ØØØ]  767:
[SF3]  768:   procedure getquell;
[HB5]  769:   var a,i,c,s,h:byte;r:registers;
[V95]  77Ø:       b:menufeld; p:^byte; label err;
[MUØ]  771:   begin
[L81]  772:   r.ah:=14;
[C22]  773:   r.dl:=orglw;
[RF1]  774:   msdos(r);
[Ø81]  775:   a:=r.al;
[WPØ]  776:   i:=2;
[ZY1]  777:   while i<=a do
[MUØ]  778:               begin
[DXØ]  779:               {$i-}
[ØX2]  78Ø:               chdir(chr(i+64)+':');
[BXØ]  781:               {$i+}
[951]  782:               inc(i);
[394]  783:               if ioresult <>Ø then a:=i-2
[ZNØ]  784:               end;
[R73]  785:   if a<2 then goto err;
[RG2]  786:   if a=3 then begin
[S81]  787:                qlw:=2;
[NX1]  788:                lwstr:='C:';
[4L1]  789:                markdirs;
[3E1]  79Ø:                goto err
[ZNØ]  791:                end;
[S71]  792:   dec(a,2);
[CV3]  793:   changedir(orgpath,nob);
[ZG2]  794:   for i:=1 to a do
[TN3]  795:       b[i]:='^'+chr(i+66)+':    ';
[BR2]  796:   textattr:=col2;
[PY2]  797:   normrahmen:=col2;
[3K4]  798:   h:=hotattr; hotattr:=hot3;
[NT6]  799:   s:=selectattribut; selectattribut:=sel2;
[SZ1]  8ØØ:   menumode:=Ø;
[CE5]  8Ø1:   infoline(' Bitte Laufwerk der zu'+
[ØF4]  8Ø2:            ' sichernden Daten angeben');
[WM3]  8Ø3:   verticalmenu(3Ø,8,a,b);
[LY1]  8Ø4:   infoline('');
[GH2]  8Ø5:   restorescreen;
[613]  8Ø6:   if not escflag then
[MUØ]  8Ø7:      begin
[UR2]  8Ø8:      qlw:=menumode+1;
[3M5]  8Ø9:      lwstr:=scanleft(b[menumode],' ');
[TR2]  81Ø:      delleft(lwstr,1)
[ZNØ]  811:      end;
[8R2]  812:   textattr:=col1;
[NY2]  813:   normrahmen:=col1;
[855]  814:   selectattribut:=s; hotattr:=h;
[AN4]  815:   if not escflag then markdirs;
[2SØ]  816:   err:
[ZNØ]  817:  end;
[ØØØ]  818:
[463]  819:  procedure getziel;
[KW4]  82Ø:  var a:menufeld; q,s,h,i:byte;
[MUØ]  821:   begin
[1R5]  822:   textattr:=col2; normrahmen:=col2;
[3K4]  823:   h:=hotattr; hotattr:=hot3;
[NT6]  824:   s:=selectattribut; selectattribut:=sel2;
[Ø12]  825:   a[1]:='^A:     ';
[612]  826:   a[2]:='^B:     ';
[Z22]  827:   q:=menumode;
[SZ1]  828:   menumode:=Ø;
[GD6]  829:   infoline(' Bitte Ziellaufwerk für die'+
[BT3]  83Ø:            ' Backup-Daten angeben');
[YK3]  831:   verticalmenu(3Ø,8,2,a);
[LN5]  832:   if not escflag then lw:=menumode-1;
[VS5]  833:   for i:=1 to 2Ø do if lwfeld[i]<>nil
[LF4]  834:     then lwfeld[i]^.zlw:=lw;
[XP5]  835:   textattr:=col1; normrahmen:=col1;
[WB5]  836:   hotattr:=h; selectattribut:=s;
[GH2]  837:   restorescreen;
[LY1]  838:   infoline('');
[122]  839:   menumode:=q
[ZNØ]  84Ø:   end;
[ØØØ]  841:
[X74]  842:   procedure getrestquell;
[8M4]  843:   var a:menufeld; q,s,h:byte;
[MUØ]  844:   begin
[1R5]  845:   textattr:=col2; normrahmen:=col2;
[3K4]  846:   h:=hotattr; hotattr:=hot3;
[NT6]  847:   s:=selectattribut; selectattribut:=sel2;
[Ø12]  848:   a[1]:='^A:     ';
[612]  849:   a[2]:='^B:     ';
[Z22]  85Ø:   q:=menumode;
[SZ1]  851:   menumode:=Ø;
[5E6]  852:   infoline(' Bitte Quellaufwerk für die'+
[BT3]  853:            ' Backup-Daten angeben');
[2L3]  854:   verticalmenu(5Ø,7,2,a);
[LY1]  855:   infoline('');
[8U5]  856:   if not escflag then rlw:=menumode-1;
[XP5]  857:   textattr:=col1; normrahmen:=col1;
[WB5]  858:   hotattr:=h; selectattribut:=s;
[GH2]  859:   restorescreen;
[122]  86Ø:   menumode:=q
```

```
[ZNØ]  861:   end;
[ØØØ]  862:
[5D3]  863:   procedure getdatum;
[GE5]  864:   var std,min,tag,mon,jahr: word;
[2G3]  865:       attr:byte; s:string;
[PV2]  866:       esc,err:boolean;
[SP4]  867:       w:string[1Ø]; date:datetime;
[ZL1]  868:       label 1,2,3;
[MUØ]  869:   begin
[DC1]  87Ø:   nomaus;
[1R5]  871:   textattr:=col2; normrahmen:=col2;
[D86]  872:   infoline(' Es werden nur Files gesich'+
[SZ3]  873:            'ert, die nach dem angege'+
[A55]  874:            'benen Datum erstellt wurden');
[3S2]  875:   rahmen(3Ø,8,27,7,'');
[DB1]  876:   clrscr;
[PX5]  877:   writeln(' Bitte Datum des letzten');
[5P4]  878:   writeln(' Backups eingeben');
[XF1]  879:   writeln;
[8R4]  88Ø:   writeln('     (Tag.Monat.Jahr)');
[WK4]  881:   writeln; gotoxy(8,wherey);
[5Ø5]  882:   textattr:=col6; write('          ');
[5Z2]  883:   gotoxy(8,wherey);
[RF5]  884:   eingabe(s,esc,1Ø); textattr:=col2;
[ZZ2]  885:   if esc then goto 2;
[DZ2]  886:   w:=scanleft(s,'.');
[BB3]  887:   tag:=wordval(w,err);
[713]  888:   if err then goto 1;
[WT3]  889:   delleft(s,length(w)+1);
[DZ2]  89Ø:   w:=scanleft(s,'.');
[3D3]  891:   mon:=wordval(w,err);
[713]  892:   if err then goto 1;
[WT3]  893:   delleft(s,length(w)+1);
[RM3]  894:   jahr:=wordval(s,err);
[713]  895:   if err then goto 1;
[DB1]  896:   clrscr;
[7L6]  897:   writeln(' Bitte Uhrzeit des letzten');
[5P4]  898:   writeln(' Backups eingeben');
[XF1]  899:   writeln;
[NU4]  9ØØ:   writeln('     (Stunde.Minute)');
[NP4]  9Ø1:   writeln; gotoxy(11,wherey);
[6R4]  9Ø2:   textattr:=col6; write('     ');
[Ø13]  9Ø3:   gotoxy(11,wherey);
[PL2]  9Ø4:   eingabe(s,esc,5);
[ZZ2]  9Ø5:   if esc then goto 2;
[DZ2]  9Ø6:   w:=scanleft(s,'.');
[PC3]  9Ø7:   std:=wordval(w,err);
[713]  9Ø8:   if err then goto 1;
[WT3]  9Ø9:   delleft(s,length(w)+1);
[5C3]  91Ø:   min:=wordval(s,err);
[713]  911:   if err then goto 1;
[U73]  912:   if tag in [1..31] then
[X83]  913:    if mon in [1..12] then
[DL2]  914:     if jahr>1979 then
[YN2]  915:      if jahr <21ØØ then
[L83]  916:       if std in [Ø..23] then
[793]  917:        if min in [Ø..59] then
[EY1]  918:          with date do
[MUØ]  919:           begin
[XY1]  92Ø:           year:=jahr;
[6Ø2]  921:           month:=mon;
[ØH1]  922:           day:=tag;
[7T1]  923:           hour:=std;
[GH1]  924:           min:=min;
[H51]  925:           sec:=Ø;
[MC4]  926:           packtime(date,searchtime);
[LZØ]  927:           goto 3
[ZNØ]  928:           end;
[G8Ø]  929:   1:
[DTØ]  93Ø:   beep;
[K8Ø]  931:   2:
[UØ4]  932:   timeflag:=not timeflag;
[M8Ø]  933:   3:
[BM1]  934:   nocursor;
[GH2]  935:   restorescreen;
[BM1]  936:   showmaus;
[LY1]  937:   infoline('');
[XP5]  938:   textattr:=col1; normrahmen:=col1;
[ZNØ]  939:   end;
[ØØØ]  94Ø:
[4P5]  941:   procedure fileex(var del:boolean);
[6X4]  942:   var a:menufeld; t,h,q,s:byte;
[MUØ]  943:   begin
[DTØ]  944:   beep;
[H64]  945:   q:=menumode;  h:=hotattr;
[BU4]  946:   hotattr:=hot1; t:=textattr;
[484]  947:   a[1]:='^nein'; a[2]:='^ja';
[5R5]  948:   textattr:=col3; normrahmen:=col3;
[5X2]  949:   rahmen(3Ø,14,28,5,'');
[DC1]  95Ø:   nomaus;
[DB1]  951:   clrscr;
[NT6]  952:   s:=selectattribut; selectattribut:=sel2;
[A46]  953:   writeln(' Datei existiert bereits !');
[HX4]  954:   writeln(' Überschreiben ?');
[BM1]  955:   showmaus;
[NT2]  956:   window(38,16,5Ø,18);
[SZ1]  957:   menumode:=Ø;
[AL3]  958:   horizontalmenu(3,2,a);
[HD5]  959:   del:=(menumode=2) and not escflag;
[3A4]  96Ø:   menumode:=q; textattr:=t;
```

```
[3V4]  961:    normrahmen:=t; textattr:=t;
[K32]  962:    hotattr:=h;
[XE5]  963:    selectattribut:=s; menumode:=q;
[ZB2]  964:    restorescreen
[ZNØ]  965:    end;
[ØØØ]  966:
[ØØØ]  967:
[WV6]  968:    procedure nottoreadwrite(flag:boolean);
[4C2]  969:    var t,ch:byte;
[MUØ]  97Ø:    begin
[872]  971:    t:=textattr;
[5R5]  972:    textattr:=col3; normrahmen:=col3;
[YW2]  973:    rahmen(19,13,42,3,'');
[DC1]  974:    nomaus;
[UT3]  975:    clrscr; writeln; beep;
[B46]  976:    if flag then write(' Konfigurations'+
[9G3]  977:                       'datei nicht lesbar !')
[DD3]  978:            else write(' Fehler'+
[UP4]  979:             ' beim Erstellen der Datei !');
[3P3]  98Ø:    infoline(' Taste oder'+
[XØ4]  981:                ' Maustaste drücken');
[BM1]  982:    showmaus;
[RG4]  983:    repeat until keypressed or
[CA3]  984:                 button1 or button2;
[9A4]  985:    if keypressed then clrbuf;
[LY1]  986:    infoline('');
[KY4]  987:    textattr:=t; normrahmen:=t;
[GH2]  988:    restorescreen;
[MG1]  989:    nobutton
[ZNØ]  99Ø:    end;
[ØØØ]  991:
[ØØØ]  992:
[DN3]  993:  procedure saveconfig;
[G74]  994:   var a,i:byte; ein:string;
[TM5]  995:   esc,ueberschr:boolean; label raus;
[MUØ]  996:   begin
[262]  997:   a:=textattr;
[1R5]  998:   textattr:=col2; normrahmen:=col2;
[AX2]  999:   rahmen(25,12,28,4,'');
[DB1] 1ØØØ:   clrscr;
[DC1] 1ØØ1:   nomaus;
[T96] 1ØØ2:   writeln(' Bitte Dateinamen eingeben');
[S65] 1ØØ3:   writeln(' (ohne Erweiterung)');
[M84] 1ØØ4:   textattr:=col6; writeln;
[945] 1ØØ5:   gotoxy(1Ø,wherey); write('        ');
[ZX5] 1ØØ6:   gotoxy(1Ø,wherey); eingabe(ein,esc,8);
[S93] 1ØØ7:   nocursor;showmaus;
[3H3] 1ØØ8:   if esc then goto raus;
[MW3] 1ØØ9:     ein:=orgpath+ein+'.mul';
[3Z2] 1Ø1Ø:     assign(datei,ein);
[DXØ] 1Ø11:     {$i-}
[F52] 1Ø12:     reset(datei);
[BXØ] 1Ø13:     {$i+}
[FØ3] 1Ø14:     if ioresult=Ø then
[MUØ] 1Ø15:       begin
[XZ2] 1Ø16:       fileex(ueberschr);
[M35] 1Ø17:       if not ueberschr then goto raus
[ZNØ] 1Ø18:       end;
[DXØ] 1Ø19:     {$i-}
[4P2] 1Ø2Ø:     rewrite(datei);
[T53] 1Ø21:     if ioresult<>Ø then
[MUØ] 1Ø22:         begin
[BW3] 1Ø23:         nottoreadwrite(false);
[FK1] 1Ø24:         goto raus
[ZNØ] 1Ø25:         end;
[MH2] 1Ø26:     for i:=1 to 2Ø do
[MUØ] 1Ø27:       begin
[1R3] 1Ø28:       if lwfeld[i]<>nil then
[W74] 1Ø29:          write(datei,lwfeld[i]^);
[T53] 1Ø3Ø:       if ioresult<>Ø then
[MUØ] 1Ø31:          begin
[BW3] 1Ø32:          nottoreadwrite(false);
[FK1] 1Ø33:          goto raus
[4MØ] 1Ø34:          end
[ZNØ] 1Ø35:       end;
[642] 1Ø36:     close(datei);
[T53] 1Ø37:     if ioresult<>Ø then
[BW3] 1Ø38:          nottoreadwrite(false);
[2WØ] 1Ø39:   raus:
[BXØ] 1Ø4Ø:   {$i+}
[DU4] 1Ø41:   textattr:=a; normrahmen:=a;
[ZB2] 1Ø42:   restorescreen
[ZNØ] 1Ø43:   end;
[ØØØ] 1Ø44:
[ØØØ] 1Ø45:
[7V6] 1Ø46:  procedure writefiles(orand,urand:byte);
[VY1] 1Ø47:  var i:byte;
[MUØ] 1Ø48:  begin
[HV1] 1Ø49:  gotoxy(2,1);
[SØ4] 1Ø5Ø:  for i:=orand to urand do
[MUØ] 1Ø51:     begin
[ZY2] 1Ø52:     gotoxy(2,wherey);
[776] 1Ø53:     if i<=dfsize then writeln(subdrs[i])
[4MØ] 1Ø54:     end
[ZNØ] 1Ø55:  end;
[ØØØ] 1Ø56:
[KL3] 1Ø57:  procedure loadconfig;
[786] 1Ø58:  var k,i,attr,t,curspos,ch,ypos,orand,
[CZ1] 1Ø59:      urand:byte;
[666] 1Ø6Ø:      fert,time,onefile,error,jh: boolean;
```

```
[1E5] 1Ø61:      s:string; zeich:char; rec:lwahl;
[HF2] 1Ø62:      label 1,weiter;
[ØØØ] 1Ø63:
[TZ2] 1Ø64:      procedure invers;
[MUØ] 1Ø65:      begin
[EV2] 1Ø66:      attr:=textattr;
[4L4] 1Ø67:      textattr:=selectattribut;
[6N2] 1Ø68:      gotoxy(2,ypos);
[N84] 1Ø69:      write(subdrs[curspos]);
[CL2] 1Ø7Ø:      textattr:=attr
[ZNØ] 1Ø71:      end;
[ØØØ] 1Ø72:
[5H3] 1Ø73:      procedure writeups;
[BY1] 1Ø74:      var a:byte;
[MUØ] 1Ø75:      begin
[262] 1Ø76:      a:=textattr;
[RR2] 1Ø77:      textattr:=col6;
[LF4] 1Ø78:      gotoxy(3,16); write(chr(24));
[UF4] 1Ø79:      gotoxy(7,16); write(chr(25));
[GV1] 1Ø8Ø:      gotoxy(1,1);
[U32] 1Ø81:      textattr:=a
[ZNØ] 1Ø82:      end;
[ØØØ] 1Ø83:
[ZX5] 1Ø84:      function infeld(s:string):boolean;
[K85] 1Ø85:      var schl:searchrec; attr:word;
[2L1] 1Ø86:          p:string;
[MUØ] 1Ø87:      begin
[8N1] 1Ø88:      dfsize:=Ø;
[7H4] 1Ø89:      findfirst(s,anyfile,schl);
[EA3] 1Ø9Ø:      while doserror=Ø do
[MUØ] 1Ø91:         begin
[KE3] 1Ø92:         if schl.attr<>$1Ø then
[793] 1Ø93:          if schl.attr<>8 then
[MUØ] 1Ø94:             begin
[1Ø2] 1Ø95:             inc(dfsize);
[1B4] 1Ø96:             with dateifeld[dfsize] do
[MUØ] 1Ø97:                begin
[4P2] 1Ø98:                name:=schl.name;
[YR2] 1Ø99:                time:=schl.time;
[EV2] 11ØØ:                attr:=schl.attr;
[VS2] 11Ø1:                size:=schl.size
[4MØ] 11Ø2:                end
[ZNØ] 11Ø3:             end;
[KD2] 11Ø4:         findnext(schl)
[ZNØ] 11Ø5:       end;
[B93] 11Ø6:      if doserror<>18 then
[Z55] 11Ø7:        infeld:=false else infeld:=true
[ZNØ] 11Ø8:      end;
[ØØØ] 11Ø9:
[MUØ] 111Ø:   begin
[XA4] 1111:   jh:=infeld(orgpath+'*.mul');
[2B3] 1112:   if not jh then goto 1;
[EH2] 1113:   if dfsize=Ø then
[MUØ] 1114:      begin
[872] 1115:      t:=textattr;
[5R5] 1116:      textattr:=col3; normrahmen:=col3;
[5W2] 1117:      rahmen(2Ø,13,4Ø,3,'');
[DC1] 1118:      nomaus;
[UT3] 1119:      clrscr; writeln; beep;
[4W5] 112Ø:      write(' Keine Konfigurationsdatei'+
[8C2] 1121:           ' vorhanden!');
[YE5] 1122:      infoline(' Taste oder Maustaste'+
[EV1] 1123:               ' drücken');
[BM1] 1124:      showmaus;
[RG4] 1125:      repeat until keypressed or
[CA3] 1126:             button1 or button2;
[9A4] 1127:      if keypressed then clrbuf;
[LY1] 1128:      infoline('');
[KY4] 1129:      textattr:=t; normrahmen:=t;
[GH2] 113Ø:      restorescreen;
[AM1] 1131:      nobutton;
[DZØ] 1132:      goto 1
[ZNØ] 1133:      end;
[RK5] 1134:   for i:=1 to dfsize do subdrs[i]:=
[G45] 1135:        scanleft(dateifeld[i].name,'.');
[FG3] 1136:   if dfsize=1 then begin
[V52] 1137:                      s:=subdrs[1];
[V32] 1138:                      goto weiter
[ZNØ] 1139:                      end;
[XP5] 114Ø:   textattr:=col1; normrahmen:=col1;
[AW2] 1141:   rahmen(3Ø,5,2Ø,16,'');
[DC1] 1142:   nomaus;
[DB1] 1143:   clrscr;
[L35] 1144:   fert:=false;  ch:=Ø; curspos:=1;
[5F5] 1145:   ypos:=1; orand:=1; onefile:=false;
[4V5] 1146:   urand:=15; writefiles(orand,urand);
[EN1] 1147:   writeups;
[2D1] 1148:   invers;
[2E6] 1149:   infoline(' Bitte Konfigurationsdatei'+
[3T1] 115Ø:            ' wählen');
[BM1] 1151:   showmaus;
[EP2] 1152:   while not fert do
[MUØ] 1153:     begin
[ØT5] 1154:     if keypressed then ch:=ord(readkey);
[YG3] 1155:     if (ch=72)or(ch=8Ø) then
[MUØ] 1156:        begin
[6N2] 1157:        gotoxy(2,ypos);
[DC1] 1158:        nomaus;
```

Listing 7. »backwins.pas« stellt grundlegende Oberflächenroutinen

```
[N84] 1159:         write(subdrs[curspos]);
[VX1] 116Ø:         if ch=8Ø then
[MUØ] 1161:            begin
[7T3] 1162:            if curspos<dfsize then
[8A2] 1163:                        inc(curspos);
[7L3] 1164:            if curspos>urand then
[MUØ] 1165:               begin
[DB1] 1166:               clrscr;
[5W1] 1167:               inc(orand);
[HW1] 1168:               inc(urand);
[914] 1169:               writefiles(orand,urand)
[4MØ] 117Ø:               end
[ZNØ] 1171:             end;
[TX1] 1172:         if ch=72 then
[MUØ] 1173:            begin
[D55] 1174:            if curspos>1 then dec(curspos);
[TK3] 1175:            if curspos<orand then
[MUØ] 1176:               begin
[DB1] 1177:               clrscr;
[ØV1] 1178:               dec(orand);
[CV1] 1179:               dec(urand);
[914] 118Ø:               writefiles(orand,urand)
[4MØ] 1181:               end
[ZNØ] 1182:            end;
[YU3] 1183:         ypos:=curspos-orand+1;
[2D1] 1184:         invers;
[EN1] 1185:         writeups;
[BM1] 1186:         showmaus;
[3RØ] 1187:         ch:=Ø
[ZNØ] 1188:         end;
[HZ3] 1189:      if (ch=27)or button2 then
[MUØ] 119Ø:         begin
[912] 1191:         fert:=true;
[ME2] 1192:         escflag:=true;
[U54] 1193:         repeat until not button2
[ZNØ] 1194:         end;
[5G2] 1195:      if button1 then
[D92] 1196:       if akty>4 then
[FE2] 1197:        if aktx>29 then
[FE2] 1198:         if aktx<39 then
[PD2] 1199:          if akty<2Ø then
[MUØ] 12ØØ:             begin
[4U3] 12Ø1:             if curspos+akty-4-ypos
[G72] 12Ø2:                <=dfsize then
[MUØ] 12Ø3:                begin
[DC1] 12Ø4:                nomaus;
[6N2] 12Ø5:                gotoxy(2,ypos);
[N84] 12Ø6:                write(subdrs[curspos]);
[R43] 12Ø7:                curspos:=curspos+
[312] 12Ø8:                         akty-4-ypos;
[W52] 12Ø9:                ypos:=akty-4;
[A31] 121Ø:                ch:=13;
[BM1] 1211:                showmaus;
[T54] 1212:                repeat until not button1
[4MØ] 1213:                end
[4MØ] 1214:              end
[G93] 1215:             else if akty=2Ø then
[VD2] 1216:                   if aktx=32 then
[MUØ] 1217:                      begin
[GC2] 1218:                      if orand>1 then
[MUØ] 1219:                         begin
[DC1] 122Ø:                         nomaus;
[DB1] 1221:                         clrscr;
[ØV1] 1222:                         dec(orand);
[CV1] 1223:                         dec(urand);
[392] 1224:                         dec(curspos);
[NW2] 1225:                         writefiles(orand,
[171] 1226:                                    urand);
[G33] 1227:                         invers; writeups;
[BM1] 1228:                         showmaus;
[1E1] 1229:                         delay(1ØØ)
[4MØ] 123Ø:                         end
[4MØ] 1231:                       end
[ZRØ] 1232:                      else
[7E2] 1233:                       if aktx=36 then
[MUØ] 1234:                          begin
[W81] 1235:                          if urand
[532] 1236:                           <dfsize then
[MUØ] 1237:                             begin
[DC1] 1238:                             nomaus;
[DB1] 1239:                             clrscr;
[5W1] 124Ø:                             inc(orand);
[HW1] 1241:                             inc(urand);
[8A2] 1242:                             inc(curspos);
[BV1] 1243:                             writefiles
[T82] 1244:                             (orand,urand);
[2D1] 1245:                             invers;
[EN1] 1246:                             writeups;
[BM1] 1247:                             showmaus;
[1E1] 1248:                             delay(1ØØ)
[4MØ] 1249:                             end
[ZNØ] 125Ø:                          end;
[VU3] 1251:      if ch=13 then fert:=true
[ZNØ] 1252:   end;
[GH2] 1253:   restorescreen;
[LY1] 1254:   infoline('');
[RN3] 1255:   if escflag then goto 1;
[UK3] 1256:   s:=subdrs[curspos];
[XA1] 1257:   weiter:
[G65] 1258:   assign(datei,orgpath+s+'.mul');
```

```
[DXØ] 1259:   {$i-}
[F52] 126Ø:   reset(datei);
[BXØ] 1261:   {$i+}
[T53] 1262:   if ioresult<>Ø then
[MUØ] 1263:      begin
[NR3] 1264:      nottoreadwrite(true);
[DZØ] 1265:      goto 1
[ZNØ] 1266:      end;
[LB5] 1267:   for i:=1 to 2Ø do lwfeld[i]:=nil;
[LT3] 1268:   while not eof(datei) do
[MUØ] 1269:     begin
[DXØ] 127Ø:     {$i-}
[8E2] 1271:     read(datei,rec);
[BXØ] 1272:     {$i+}
[T53] 1273:     if ioresult<>Ø then
[MUØ] 1274:        begin
[NR3] 1275:        nottoreadwrite(true);
[DZØ] 1276:        goto 1
[ZNØ] 1277:        end;
[VH3] 1278:     zeich:=rec.sub[1][1];
[Y42] 1279:     lw:=rec.zlw;
[VV3] 128Ø:     i:=ord(upchar(zeich))-65;
[EP5] 1281:     new(lwfeld[i]); lwfeld[i]^:=rec
[ZNØ] 1282:     end;
[G8Ø] 1283:   1:
[ZNØ] 1284:   end;
[ØØØ] 1285:
[TD6] 1286:   procedure writes(lw,start,zahl:word;
[ZZ4] 1287:             p:pointer; var error:boolean);
[6S2] 1288:   var count:byte;
[MUØ] 1289:   begin
[7G1] 129Ø:   count:=2;
[P31] 1291:   repeat
[AA6] 1292:   writesektor(lw,start,zahl,p^,error);
[3P1] 1293:   dec(count)
[CN4] 1294:   until not error or (count=Ø)
[ZNØ] 1295:   end;
[ØØØ] 1296:
[G76] 1297:   procedure reads(lw,start,zahl:word;
[ZZ4] 1298:             p:pointer; var error:boolean);
[6S2] 1299:   var count:byte;
[MUØ] 13ØØ:   begin
[8G1] 13Ø1:   count:=3;
[P31] 13Ø2:   repeat
[776] 13Ø3:   readsektor(lw,start,zahl,p^,error);
[3P1] 13Ø4:   dec(count)
[CN4] 13Ø5:   until not error or (count=Ø)
[ZNØ] 13Ø6:   end;
[ØØØ] 13Ø7:
[ØØØ] 13Ø8:
[MUØ] 13Ø9: begin
[LB5] 131Ø:   for i:=1 to 2Ø do lwfeld[i]:=nil;
[263] 1311:   if color then begin
[ER3] 1312:                 col1:=blue shl 4+white;
[6Ø3] 1313:                 col2:=green shl 4;
[PH3] 1314:                 col3:=red shl 4+white;
[4K2] 1315:                 col4:=cyan shl 4;
[384] 1316:                 col5:=magenta shl 4+white;
[542] 1317:                 col6:=white;
[SK4] 1318:                 col7:=magenta shl 4+yellow;
[223] 1319:                 col8:=brown shl 4;
[3K2] 132Ø:                 sel1:=cyan shl 4;
[342] 1321:                 sel2:=white;
[HX3] 1322:                 hot1:=red shl 4+yellow;
[FY3] 1323:                 hot2:=blue shl 4+yellow;
[7P3] 1324:                 hot3:=green shl 4+red;
[4MØ] 1325:                 end
[WP1] 1326:            else begin
[BK2] 1327:                 col1:=4; col2:=4;
[HK2] 1328:                 col3:=4; col4:=4;
[VX2] 1329:                 col5:=4; col6:=$7Ø;
[YT2] 133Ø:                 col7:=$7Ø; col8:=4;
[4W2] 1331:                 sel1:=$7Ø; sel2:=$7Ø;
[RX2] 1332:                 hot1:=15; hot2:=15;
[491] 1333:                 hot3:=15
[ZNØ] 1334:                 end;
[RW3] 1335:   getintvec($1b,oldbreak);
[L24] 1336:   setintvec($1b,@neubreak);
[9C1] 1337:   r.ah:=$19;
[RF1] 1338:   msdos(r);
[812] 1339:   orglw:=r.al;
[VØ3] 134Ø:   getdir(Ø,orgpath);
[8L4] 1341:   if right(orgpath,1)<>'\' then
[BH3] 1342:          orgpath:=orgpath+'\'
[KNØ] 1343:   end.
```

Listing 7. »backwins.pas« stellt grundlegende Oberflächenroutinen

```
[S84]    1: { Programm: diskform.pas
[X56]    2:   Funktion: Formatieren von Disketten
[8Z5]    3:   Sprache:  Turbo Pascal ab Version 5.5
[R94]    4:   Autor:    Frank Emminghaus
[P35]    5:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    6:
[DV2]    7: unit diskformat;
[ØØØ]    8:
[SK1]    9: interface
[BP3]   1Ø: uses crt,dos,windows;
```

```
[ØØØ]   11:
[STØ]   12: type
[636]   13:  boot = record spr:array[Ø..2] of byte;
[NN2]   14:                vol:string[7];
[3L2]   15:                sektsize:word;
[CT2]   16:                clustsize:byte;
[FL2]   17:                bootsize:word;
[R42]   18:                fatcop:byte;
[4S2]   19:                hauptsize: word;
[3S2]   2Ø:                disksize: word;
[ØG1]   21:                id: byte;
[CF2]   22:                fatsize: word;
[AN2]   23:                spursize:word;
[ØZ1]   24:                sides: word;
[LL4]   25:                rest:array[28..511] of byte
[ZNØ]   26:                end;
[ØØØ]   27:
[RNØ]   28: var
[L92]   29:   bootsec:boot;
[ØØØ]   3Ø:
[D95]   31: procedure getdrivepar(lw:byte;
[BH2]   32:                           var typ:byte);
[8G6]   33: procedure setpar(lw,typ,format:byte;
[HB3]   34:                  var error:boolean);
[SP5]   35: procedure format(lw,spuren:byte;
[HB3]   36:                  var error:boolean);
[ØØØ]   37:
[ØØØ]   38:
[4L2]   39: implementation
[ØØØ]   4Ø:
[ML5]   41: type formats=record spur,seite,
[ØS2]   42:                     sektor,bps:byte
[ZNØ]   43:                     end;
[ØØØ]   44:
[6XØ]   45: const
[1D5]   46:  bootblock:array[Ø..144] of byte =
[L34]   47:      ($eb,$1a,$9Ø,$2Ø,$2Ø,$2Ø,$2Ø,$2Ø,$2Ø,
[114]   48:       $2Ø,$2Ø,$ØØ,$Ø2,$Ø2,$Ø1,$ØØ,$Ø2,$7Ø,
[964]   49:       $ØØ,$aØ,$Ø5,$f9,$Ø3,$ØØ,$Ø9,$ØØ,$Ø2,
[3K4]   5Ø:       $ØØ,$fa,$33,$cØ,$8e,$dØ,$bc,$ØØ,$7c,
[5T4]   51:       $fb,$fc,$b8,$cØ,$Ø7,$8e,$d8,$be,$48,
[2G4]   52:       $ØØ,$b4,$Øe,$ac,$22,$cØ,$74,$Ø4,$cd,
[PL4]   53:       $1Ø,$eb,$f7,$32,$e4,$cd,$16,$b8,$Ød,
[UL4]   54:       $Øe,$cd,$1Ø,$bØ,$Øa,$cd,$1Ø,$cd,$19,
[EB4]   55:       $Ød,$Øa,$44,$69,$65,$73,$2Ø,$69,$73,
[KB4]   56:       $74,$2Ø,$6b,$65,$69,$6e,$65,$2Ø,$53,
[YD4]   57:       $79,$73,$74,$65,$6d,$64,$69,$73,$6b,
[XB4]   58:       $65,$74,$74,$65,$Ød,$Øa,$42,$69,$74,
[X94]   59:       $74,$65,$2Ø,$65,$72,$73,$65,$74,$7a,
[9C4]   6Ø:       $65,$6e,$2Ø,$75,$6e,$64,$2Ø,$65,$69,
[X84]   61:       $6e,$65,$2Ø,$54,$61,$73,$74,$65,$2Ø,
[RF4]   62:       $64,$72,$81,$63,$6b,$65,$6e,$2e,$2e,
[HMØ]   63:       $2e);
[ØØØ]   64:
[ØØØ]   65:
[ØØØ]   66:
[E96]   67: var lesepuffer:array[Ø..9215] of byte;
[A96]   68:     spurformat: array[1..18] of formats;
[RM2]   69:     spursize:byte;
[Ø43]   7Ø:     lwseg,lwtab: word;
[ØØØ]   71:
[D95]   72: procedure getdrivepar(lw:byte;
[BH2]   73:                           var typ:byte);
[6Y2]   74: var r:registers;
[MUØ]   75: begin
[Ø61]   76:  r.ah:=8;
[KD1]   77:  r.dl:=lw;
[YR1]   78:  intr($13,r);
[FH1]   79:  typ:=r.bl
[ZNØ]   8Ø: end;
[ØØØ]   81:
[8G6]   82: procedure setpar(lw,typ,format:byte;
[HB3]   83:                  var error:boolean);
[564]   84: var r:registers; label 1;
[MUØ]   85: begin
[4R3]   86:  fillchar(bootsec,512,Ø);
[LK4]   87:  move(bootblock,bootsec,145);
[7C1]   88:  r.ah:=$18;
[KD1]   89:  r.dl:=lw;
[RD2]   9Ø:  case format of
[351]   91:   1: begin
[WM1]   92:      r.cx:=$27Ø9;
[3G3]   93:      mem[lwseg:lwtab]:=9;
[CC3]   94:      bootsec.fatsize:=2;
[SM2]   95:      bootsec.id:=$fd;
[ØØ4]   96:      bootsec.hauptsize:=$7Ø;
[HL3]   97:      bootsec.disksize:=72Ø
[ZNØ]   98:      end;
[551]   99:   2: begin
[GR1]  1ØØ:      r.cx:=$4fØf;
[LK3]  1Ø1:      mem[lwseg:lwtab]:=15;
[LN3]  1Ø2:      bootsec.clustsize:=1;
[VC3]  1Ø3:      bootsec.fatsize:=7;
[WØ4]  1Ø4:      bootsec.hauptsize:=$eØ;
[VY3]  1Ø5:      bootsec.disksize:=24ØØ;
[YH3]  1Ø6:      bootsec.spursize:=15
[ZNØ]  1Ø7:      end;
[GB3]  1Ø8:   3: if typ=2 then begin
[ØX2]  1Ø9:                    mem[Ø:$49Ø+lw]:=$54;
[3G3]  11Ø:                    mem[lwseg:lwtab]:=9;
[G82]  111:                    error:=false;
```

```
[DZØ]  112:                    goto 1
[4MØ]  113:                    end
[WP1]  114:               else begin
[WN1]  115:                    r.cx:=$4fØ9;
[8E3]  116:                    mem[lwseg:lwtab]:=9
[ZNØ]  117:                    end;
[951]  118:   4: begin
[3N1]  119:      r.cx:=$4f12;
[1D3]  12Ø:      bootsec.fatsize:=9;
[PK3]  121:      mem[lwseg:lwtab]:=18;
[LN3]  122:      bootsec.clustsize:=1;
[WØ4]  123:      bootsec.hauptsize:=$eØ;
[FZ3]  124:      bootsec.disksize:=288Ø;
[1K3]  125:      bootsec.spursize:=18
[ZNØ]  126:      end;
[ZNØ]  127:   end;
[YR1]  128:   intr($13,r);
[ES5]  129:   error:=boolean(r.flags and fcarry);
[G8Ø]  13Ø:   1:
[ZNØ]  131: end;
[ØØØ]  132:
[ØØØ]  133:
[567]  134: procedure formatspur(lw,spur,seite:byte;
[HB3]  135:                      var error:boolean);
[D94]  136: var i:word; r:registers;
[MUØ]  137: begin
[G44]  138:  for i:=1 to spursize do
[MUØ]  139:    begin
[MP4]  14Ø:    spurformat[i].spur:=spur;
[135]  141:    spurformat[i].seite:=seite;
[WF4]  142:    spurformat[i].sektor:=i;
[LM3]  143:    spurformat[i].bps:=2
[ZNØ]  144:    end;
[M51]  145:  r.ah:=5;
[9R2]  146:  r.al:=spursize;
[2R3]  147:  r.bx:=ofs(spurformat);
[ØR3]  148:  r.es:=seg(spurformat);
[NY1]  149:  r.ch:=spur;
[G51]  15Ø:  r.cl:=1;
[ZØ2]  151:  r.dh:=seite;
[KD1]  152:  r.dl:=lw;
[YR1]  153:  intr($13,r);
[EX4]  154:  if (r.flags and fcarry)<>Ø then
[MUØ]  155:                              begin
[352]  156:                              error:=true;
[ASØ]  157:                              exit
[ZNØ]  158:                              end;
[851]  159:  r.ah:=2;
[9R2]  16Ø:  r.al:=spursize;
[ZM3]  161:  r.bx:=ofs(lesepuffer);
[XM3]  162:  r.es:=seg(lesepuffer);
[NY1]  163:  r.ch:=spur;
[G51]  164:  r.cl:=1;
[ZØ2]  165:  r.dh:=seite;
[KD1]  166:  r.dl:=lw;
[YR1]  167:  intr($13,r);
[EX4]  168:  if (r.flags and fcarry)<>Ø then
[MUØ]  169:                              begin
[352]  17Ø:                              error:=true;
[ASØ]  171:                              exit
[ZNØ]  172:                              end;
[U42]  173:  error:=false
[ZNØ]  174:  end;
[ØØØ]  175:
[ØØØ]  176:
[SP5]  177:  procedure format(lw,spuren:byte;
[HB3]  178:                   var error:boolean);
[TR5]  179:  var counter,i:word; err:boolean;
[F72]  18Ø:      r:registers;
[MUØ]  181:  begin
[Ø65]  182:  spursize:=mem[lwseg:lwtab];
[Ø51]  183:  r.ah:=Ø;
[KD1]  184:  r.dl:=lw;
[YR1]  185:  intr($13,r);
[UPØ]  186:  i:=Ø;
[KD5]  187:  while (i<spuren)and not error do
[MUØ]  188:    begin
[DC1]  189:    nomaus;
[ZY2]  19Ø:    gotoxy(2,wherey);
[M94]  191:    write('Spur: ',i,' Seite: Ø');
[BM1]  192:    showmaus;
[DØ2]  193:    counter:=3;
[P31]  194:    repeat
[BW3]  195:    formatspur(lw,i,Ø,err);
[842]  196:    dec(counter)
[2N4]  197:    until not err or (counter=Ø);
[PC2]  198:    if not err then
[MUØ]  199:       begin
[DC1]  2ØØ:       nomaus;
[ZY2]  2Ø1:       gotoxy(2,wherey);
[S94]  2Ø2:       write('Spur: ',i,' Seite: 1');
[BM1]  2Ø3:       showmaus;
[DØ2]  2Ø4:       counter:=3;
[P31]  2Ø5:       repeat
[DW3]  2Ø6:       formatspur(lw,i,1,err);
[Y72]  2Ø7:       dec(counter);
[2G4]  2Ø8:       until not err or(counter=Ø)
[ZNØ]  2Ø9:       end;
```

Listing 8. Der Quelltext »diskform.pas« der Unit dient zum Formatieren von Disketten

```
[812]  210:     error:=err;
[XXØ]  211:     inc(i)
[ZNØ]  212:     end;
[PR2]  213:   if not error then
[MUØ]  214:     begin
[DC1]  215:     nomaus;
[ZY2]  216:     gotoxy(2,wherey);
[Y84]  217:     write('Systembereich...  ');
[BM1]  218:     showmaus;
[RK1]  219:     r.ax:=$3Ø1;
[863]  220:     r.bx:=ofs(bootsec);
[663]  221:     r.es:=seg(bootsec);
[W51]  222:     r.cx:=1;
[YD1]  223:     r.dx:=lw;
[YR1]  224:     intr($13,r);
[ES5]  225:     error:=boolean(r.flags and fcarry);
[M63]  226:     if error then exit;
[SK1]  227:     r.ax:=$4Ø1;
[863]  228:     r.bx:=ofs(bootsec);
[663]  229:     r.es:=seg(bootsec);
[W51]  230:     r.cx:=1;
[YD1]  231:     r.dx:=lw;
[YR1]  232:     intr($13,r);
[2H5]  233:     error:=boolean(r.flags and fcarry)
[4MØ]  234:     end
[ZNØ]  235:  end;
[ØØØ]  236:
[MUØ]  237: begin
[D93]  238: lwseg:=memw[Ø:$7a];
[VB3]  239: lwtab:=memw[Ø:$78]+4;
[SG4]  240: move(bootblock,bootsec,145)
[KNØ]  241: end.
```

Listing 8. Der Quelltext »diskform.pas« der Unit dient zum Formatieren von Disketten

```
[6U3]    1: { Programm: dosadd.pas
[5G5]    2:   Funktion: Unit für Datei-Handles
[8Z5]    3:   Sprache:  Turbo Pascal ab Version 5.5
[R94]    4:   Autor:    Frank Emminghaus
[P35]    5:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    6:
[Ø22]    7: unit dosadd;
[SK1]    8: interface
[ØØØ]    9:
[2M6]   1Ø: function open(path:string; modus:word;
[A94]   11:                var error:boolean): word;
[UC6]   12: function make(path:string; attr:word;
[354]   13:                var error:boolean):word;
[9U5]   14: function writeh(handle,zahl:word;
[Ø22]   15:                 var puffer;
[354]   16:                 var error:boolean):word;
[RF5]   17: function readh(handle,zahl:word;
[Ø22]   18:                var puffer;
[354]   19:                var error:boolean):word;
[RØ5]   2Ø: procedure closeh(handle:word;
[HB3]   21:                  var error:boolean);
[WM7]   22: procedure setpoi(handle:word; zahl:longint;
[HB3]   23:                  var error:boolean);
[XX4]   24: function getpoi(handle:word;
[AT4]   25:                 var error:boolean):longint;
[Z65]   26: procedure deleteh(path:string;
[HB3]   27:                   var error:boolean);
[1S7]   28: procedure writesektor(lw,start,anzahl:word;
[Ø22]   29:                       var puffer;
[HB3]   3Ø:                       var error:boolean);
[WE7]   31: procedure readsektor(lw,start,anzahl:word;
[Ø22]   32:                      var puffer;
[HB3]   33:                      var error:boolean);
[6U5]   34: procedure kopiere(var ziel,quelle;
[B82]   35:                   anzahl:word);
[ØØØ]   36:
[ØØØ]   37:
[4L2]   38: implementation
[2M6]   39: function open(path:string; modus:word;
[A94]   4Ø:                var error:boolean): word;
[VL1]   41:                external;
[UC6]   42: function make(path:string; attr:word;
[354]   43:                var error:boolean):word;
[VL1]   44:                external;
[9U5]   45: function writeh(handle,zahl:word;
[Ø22]   46:                 var puffer;
[354]   47:                 var error:boolean):word;
[VL1]   48:                 external;
[RF5]   49: function readh(handle,zahl:word;
[Ø22]   5Ø:                var puffer;
[354]   51:                var error:boolean):word;
[VL1]   52:                external;
[RØ5]   53: procedure closeh(handle:word;
[HB3]   54:                  var error:boolean);
[VL1]   55:                  external;
[WM7]   56: procedure setpoi(handle:word; zahl:longint;
[HB3]   57:                  var error:boolean);
[VL1]   58:                  external;
[XX4]   59: function getpoi(handle:word;
[AT4]   6Ø:                 var error:boolean):longint;
[VL1]   61:                 external;
[Z65]   62: procedure deleteh(path:string;
```

```
[HB3]   63:                   var error:boolean);
[VL1]   64:                   external;
[1S7]   65: procedure writesektor(lw,start,anzahl:word;
[Ø22]   66:                       var puffer;
[HB3]   67:                       var error:boolean);
[VL1]   68:                       external;
[WE7]   69: procedure readsektor(lw,start,anzahl:word;
[Ø22]   7Ø:                      var puffer;
[HB3]   71:                      var error:boolean);
[VL1]   72:                      external;
[6U5]   73: procedure kopiere(var ziel,quelle;
[P14]   74:                   anzahl:word); external;
[ØØØ]   75:
[W72]   76: {$L dosadd}
[ØØØ]   77:
[KNØ]   78: end.
```

Listing 9. In »dosadd.pas« sind die Schnittstellen zur Dateiverwaltung festgelegt

```
[MG3]    1: { Programm: maus.pas
[SN5]    2:   Funktion: Unit zur Maussteuerung
[8Z5]    3:   Sprache:  Turbo Pascal ab Version 5.5
[R94]    4:   Autor:    Frank Emminghaus
[P35]    5:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    6:
[3R1]    7: unit maus;
[ØØØ]    8:
[SK1]    9:  interface
[ØØØ]   1Ø:
[KD3]   11:  type eventhandler =
[FU5]   12:       procedure(ereignis,ButtonStatus,
[FV2]   13:                 xpos,ypos:word);
[ØØØ]   14:
[ZF3]   15:  procedure mausinit;
[E53]   16:  procedure mausend;
[MT3]   17:  procedure showcursor;
[XN3]   18:  procedure hidecursor;
[E35]   19:  function  getmask(screenmaske,
[ØR4]   2Ø:                cursormaske:word):longint;
[2R6]   21:  function  maske(zeichen:char; attribut,
[FG3]   22:                  modus:byte): longint;
[876]   23:  procedure mausmaske(maske:longint);
[1N6]   24:  procedure mausbereich(x1,y1,x2,y2: word);
[CØ7]   25:  procedure mausspeed(xSpeed,ySpeed: word);
[NG6]   26:  procedure setcursor(xpos,ypos:word);
[LN5]   27:  procedure installhandler(handler:
[NC2]   28:                           eventhandler);
[YV4]   29:  procedure deinstallhandler;
[ØØØ]   3Ø:
[6XØ]   31: const
[CX1]   32:  noevent = Ø;
[622]   33:  mausmove = 1;
[MR2]   34:  button1press = 2;
[MF2]   35:  button1rel   = 4;
[YR2]   36:  button2press = 8;
[WM2]   37:  button2rel   = 16;
[5E3]   38:  gleichzeichen = $ØØff;
[L73]   39:  gleichfarbe   = $ØØff;
[3C3]   4Ø:  invertfarbe   = $7777;
[ØØØ]   41:
[RNØ]   42: var
[6T4]   43:  mausavail,handlerinstalled,
[2S2]   44:  grafik: boolean;
[ZC1]   45:  button1,
[ZE3]   46:  button2      : boolean;
[A15]   47:  aktx, akty, aktbuttons : word;
[WA4]   48:  handlervar: eventhandler;
[ED3]   49:  ereignismaske: word;
[ØØØ]   5Ø:
[ØØØ]   51:
[4L2]   52: implementation
[VG1]   53: uses dos;
[GN1]   54:  {$l maus}
[ØØØ]   55:
[XK5]   56: type mask = record lo,hi: word end;
[XP4]   57:      wrec = record lo,hi: byte end;
[ØØØ]   58:
[RNØ]   59: var
[ZZ2]   6Ø:  exitold: pointer;
[ØØØ]   61:
[ZWØ]   62:  {$f+}
[ØV5]   63:  procedure firsthandler; external;
[1XØ]   64:  {$f-}
[ØØØ]   65:
[ZF3]   66:  procedure mausinit;
[TU2]   67:  var r: registers;
[MUØ]   68:  begin
[Z84]   69:  handlerinstalled:= false;
[ØØØ]   7Ø:
[G51]   71:  r.ax:=Ø;
[6S1]   72:  intr($33,r);
[2U3]   73:  mausavail:=(r.ax=$ffff);
[ØØØ]   74:
[VU2]   75:  if mausavail then
[MUØ]   76:        begin
[191]   77:        r.ax:=12;
[VD1]   78:        r.cx:=$1f;
```

```
[Ø24]   79:         r.dx:= ofs(firsthandler);
[XØ4]   8Ø:         r.es:= seg(firsthandler);
[BP1]   81:         intr($33,r)
[4MØ]   82:         end
[ZNØ]   83:  end;
[ØØØ]   84:
[ZWØ]   85: {$f+}
[E53]   86:  procedure mausend;
[6Y2]   87:  var r:registers;
[MUØ]   88:  begin
[Ø32]   89:  hidecursor;
[191]   9Ø:  r.ax:=12;
[S51]   91:  r.cx:=Ø;
[6S1]   92:  intr($33,r);
[G51]   93:  r.ax:=Ø;
[6S1]   94:  intr($33,r);
[XD3]   95:  exitproc := exitold
[ZNØ]   96:  end;
[1XØ]   97: {$f-}
[ØØØ]   98:
[MT3]   99:  procedure showcursor;
[6Y2]  1ØØ:  var r:registers;
[MUØ]  1Ø1:  begin
[M51]  1Ø2:  r.ax:=1;
[BP1]  1Ø3:  intr($33,r)
[ZNØ]  1Ø4:  end;
[ØØØ]  1Ø5:
[XN3]  1Ø6:  procedure hidecursor;
[6Y2]  1Ø7:  var r:registers;
[MUØ]  1Ø8:  begin
[S51]  1Ø9:  r.ax:=2;
[BP1]  11Ø:  intr($33,r)
[ZNØ]  111:  end;
[ØØØ]  112:
[ØØØ]  113:
[DE6]  114:  function maske(zeichen:char; attribut,
[1G3]  115:                 modus:byte):longint;
[AH4]  116:  var sm,cm: word; p:longint;
[MUØ]  117:  begin
[Z32]  118:  case modus of
[151]  119:  Ø: begin
[WD1]  12Ø:     sm:=$77ØØ;
[SC4]  121:     wrec(cm).lo:=ord(zeichen);
[WF2]  122:     wrec(cm).hi:=$77
[ZNØ]  123:     end;
[351]  124:  1: begin
[3YØ]  125:     sm:=Ø;
[SC4]  126:     wrec(cm).lo:=ord(zeichen);
[8L3]  127:     wrec(cm).hi:=attribut
[ZNØ]  128:     end;
[551]  129:  2: begin
[WD1]  13Ø:     sm:=$77ØØ;
[SC4]  131:     wrec(cm).lo:=ord(zeichen);
[D82]  132:     wrec(cm).hi:=Ø
[ZNØ]  133:     end;
[751]  134:  3: begin
[YF1]  135:     sm:=$77FF;
[B71]  136:     cm:=$77ØØ
[4MØ]  137:     end
[ZNØ]  138:  end;
[9G2]  139:  mask(p).lo:=cm;
[FG2]  14Ø:  mask(p).hi:=sm;
[GB1]  141:  maske:=p
[ZNØ]  142:  end;
[ØØØ]  143:
[ØØØ]  144:
[4A7]  145:  function getmask(screenmaske,cursormaske:
[2F2]  146:                   word):longint;
[ØF2]  147:  var p:longint;
[MUØ]  148:  begin
[H24]  149:  mask(p).hi:=screenmaske;
[F54]  15Ø:  mask(p).lo:=cursormaske;
[5U1]  151:  getmask:=p
[ZNØ]  152:  end;
[ØØØ]  153:
[876]  154:  procedure mausmaske(maske:longint);
[6Y2]  155:  var r:registers;
[MUØ]  156:  begin
[Z81]  157:  r.ax:=1Ø;
[M51]  158:  r.bx:=Ø;
[DA3]  159:  r.cx:=mask(maske).hi;
[7B3]  16Ø:  r.dx:=mask(maske).lo;
[BP1]  161:  intr($33,r)
[ZNØ]  162:  end;
[ØØØ]  163:
[ØØØ]  164:
[1N6]  165:  procedure mausbereich(x1,y1,x2,y2: word);
[6Y2]  166:  var r:registers;
[MUØ]  167:  begin
[C61]  168:  r.ax:=7;
[GR3]  169:  if not grafik then begin
[SD2]  17Ø:                    r.cx:=x1 shl 3;
[H62]  171:                    r.dx:=x2 shl 3
[4MØ]  172:                    end
[WP1]  173:               else begin
[4E1]  174:                    r.cx:=x1;
[LA1]  175:                    r.dx:=x2
[ZNØ]  176:                    end;
[6S1]  177:  intr($33,r);
[G61]  178:  r.ax:=8;
```

```
[GR3]  179:  if not grafik then begin
[WD2]  18Ø:                    r.cx:=y1 shl 3;
[N62]  181:                    r.dx:=y2 shl 3
[4MØ]  182:                    end
[WP1]  183:               else begin
[8E1]  184:                    r.cx:=y1;
[RA1]  185:                    r.dx:=y2
[ZNØ]  186:                    end;
[BP1]  187:  intr($33,r)
[ZNØ]  188:  end;
[ØØØ]  189:
[CZ6]  19Ø:  procedure mausspeed(xSpeed,ySpeed:word);
[6Y2]  191:  var r:registers;
[MUØ]  192:  begin
[181]  193:  r.ax:=$f;
[D52]  194:  r.cx:=xSpeed;
[N52]  195:  r.dx:=ySpeed;
[BP1]  196:  intr($33,r)
[ZNØ]  197:  end;
[ØØØ]  198:
[NG6]  199:  procedure setcursor(xpos,ypos:word);
[6Y2]  2ØØ:  var r:registers;
[MUØ]  2Ø1:  begin
[Ø61]  2Ø2:  r.ax:=4;
[932]  2Ø3:  aktx:=xpos;
[E32]  2Ø4:  akty:=ypos;
[843]  2Ø5:  if grafik then begin
[GZ1]  2Ø6:                  r.cx:=xpos;
[CS1]  2Ø7:                  r.dx:=ypos
[4MØ]  2Ø8:                  end
[WP1]  2Ø9:             else begin
[ML2]  21Ø:                  r.cx:=xpos shl 3;
[6G2]  211:                  r.dx:=ypos shl 3
[ZNØ]  212:                  end;
[BP1]  213:  intr($33,r)
[ZNØ]  214:  end;
[ØØØ]  215:
[LC7]  216:  procedure maushandler(event,buttonstatus,
[FV2]  217:                        xpos,ypos:word);
[MUØ]  218:  begin
[843]  219:  if grafik then begin
[932]  22Ø:                  aktx:=xpos;
[2W1]  221:                  akty:=ypos
[4MØ]  222:                  end
[WP1]  223:             else begin
[ZR2]  224:                  aktx:=xpos shr 3;
[EM2]  225:                  akty:=ypos shr 3
[ZNØ]  226:                  end;
[ØØØ]  227:
[YP4]  228:  aktbuttons:=buttonstatus;
[266]  229:  button1:=boolean(buttonstatus and 1);
[A66]  23Ø:  button2:=boolean(buttonstatus and 2);
[ØØØ]  231:
[ØN6]  232:  if boolean (event and ereignismaske) then
[H24]  233:      if handlerinstalled then
[SA5]  234:              handlervar(event,buttonstatus,
[E12]  235:                         xpos,ypos);
[ØØØ]  236:
[ZNØ]  237:  end;
[ØØØ]  238:
[LN5]  239:  procedure installhandler(handler:
[NC2]  24Ø:                           eventhandler);
[MUØ]  241:  begin
[AH3]  242:  handlervar:=handler;
[7Ø4]  243:  handlerinstalled:=true
[ZNØ]  244:  end;
[ØØØ]  245:
[YV4]  246:  procedure deinstallhandler;
[MUØ]  247:  begin
[A74]  248:  handlerinstalled:=false
[ZNØ]  249:  end;
[ØØØ]  25Ø:
[MUØ]  251: begin
[EL1]  252: mausinit;
[E93]  253: exitold:=exitproc;
[XF3]  254: exitproc:=@mausend;
[AN2]  255: button1:=false;
[EN2]  256: button2:=false;
[LB2]  257: grafik:=false;
[KW2]  258: ereignismaske:=Ø;
[MA2]  259: aktbuttons:=Ø
[KNØ]  26Ø: end.
```

Listing 10. »maus.pas« ist eine Turbo-Pascal-Unit zur Maussteuerung

```
[U94]    1: { Programm: strings.pas
[5E5]    2:   Funktion: Aufrufe der Assembler-
[ZM4]    3:             Stringbehandlungsroutinen
[8Z5]    4:   Sprache:  Turbo Pascal ab Version 5.5
[R94]    5:   Autor:    Frank Emminghaus
[P35]    6:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    7:
[8A2]    8: unit strings;
[ØØØ]    9:
[SK1]   1Ø: interface
[9X5]   11: type posfeld=array[Ø..255] of byte;
```

Listing 11. »strings.pas« enthält Pascal-Aufrufe der Assembler-String-Behandlungsroutinen

```
[ØØØ]   12:
[ZN5]   13: function asstring(var s):string;
[V36]   14: function astype(s:string):pointer;
[BM5]   15: procedure alstyp(s:string; var f);
[764]   16: function left(s:string;
[ØM3]   17:                anzahl:byte):string;
[N84]   18: function right(s:string;
[ØM3]   19:                anzahl:byte):string;
[1W4]   2Ø: function lochar(c:char):char;
[ØX4]   21: function upchar(c:char):char;
[LC6]   22: function upstring(s:string):string;
[MB6]   23: function lostring(s:string):string;
[YC6]   24: function prefix(s,p:string):boolean;
[MD6]   25: function suffix(s,p:string):boolean;
[W96]   26: function substring(s,p:string):byte;
[9F4]   27: function intval(s:string;
[NS4]   28:          var error:boolean):integer;
[LM4]   29: function wordval(s:string;
[354]   3Ø:          var error:boolean):word;
[AF6]   31: procedure delleft(var s; anzahl:byte);
[4G6]   32: procedure delright(var s; anzahl:byte);
[P56]   33: function intstr(i: integer):string;
[2T5]   34: function wordstr(i:word):string;
[9X4]   35: function scanleft(s:string;
[FT2]   36:                   ch:char):string;
[874]   37: function copy(s:string;
[B43]   38:               p1,p2:byte):string;
[RZ4]   39: function scanright(s:string;
[FT2]   4Ø:                   ch:char):string;
[G15]   41: function vorkommen(s:string;
[AX2]   42:                   ch:char):pointer;
[YP6]   43: procedure getposfeld(s:string; ch:char;
[VL2]   44:                      var p:posfeld);
[4L2]   45: implementation
[ØØØ]   46:
[4E7]   47: function asstring(var s):string; external;
[6M4]   48: function astype(s:string):
[Y73]   49:          pointer; external;
[BM5]   5Ø: procedure alstyp(s:string; var f);
[VL1]   51:           external;
[TF6]   52: function left(s:string; anzahl:byte):
[B13]   53:          string; external;
[KM6]   54: function right(s:string; anzahl:byte):
[B13]   55:          string; external;
[CN6]   56: function lochar(c:char):char; external;
[BP6]   57: function upchar(c:char):char; external;
[3Ø5]   58: function upstring(s:string):
[B13]   59:          string; external;
[4Z4]   6Ø: function lostring(s:string):
[B13]   61:          string; external;
[5Z4]   62: function prefix(s,p:string):
[Y53]   63:          boolean; external;
[VZ4]   64: function suffix(s,p:string):
[Y53]   65:          boolean; external;
[XF5]   66: function substring(s,p:string):
[5P2]   67:          byte; external;
[ZA6]   68: function intval(s:string; var error:
[8R4]   69:          boolean):integer; external;
[2A6]   7Ø: function wordval(s:string; var error:
[844]   71:          boolean):word; external;
[AF6]   72: procedure delleft(var s; anzahl:byte);
[VL1]   73:           external;
[4G6]   74: procedure delright(var s; anzahl:byte);
[VL1]   75:           external;
[P56]   76: function intstr(i: integer):string;
[VL1]   77:          external;
[7H7]   78: function wordstr(i:word):string; external;
[AC6]   79: function scanleft(s:string; ch:char):
[B13]   8Ø:          string; external;
[3Ø6]   81: function copy(s:string; p1,p2:byte):
[B13]   82:          string; external;
[EG6]   83: function scanright(s:string; ch:char):
[B13]   84:          string; external;
[7K6]   85: function vorkommen(s:string; ch:char):
[Y73]   86:          pointer; external;
[YP6]   87: procedure getposfeld(s:string; ch:char;
[P84]   88:           var p:posfeld); external;
[ØØØ]   89:
[Z82]   9Ø: {$L strings}
[ØØØ]   91:
[KNØ]   92: end.
```

Listing 11. »strings.pas« enthält Pascal-Aufrufe der Assembler-String-Behandlungsroutinen

```
[Y94]    1: { Programm: windows.pas
[5E5]    2:   Funktion: Aufrufe der Assembler-
[4Z4]    3:             Fenster- und Menübehandlung
[8Z5]    4:   Sprache:  Turbo Pascal ab Version 5.5
[R94]    5:   Autor:    Frank Emminghaus
[P35]    6:   (c)1991 DMV Widuch GmbH & Co. KG }
[ØØØ]    7:
[XA2]    8: unit windows;
[SK1]    9: interface
[TG4]   1Ø: uses crt,dos,maus,strings;
[ØØØ]   11:
[STØ]   12: type
[W56]   13:   menufeld=array[1..15] of string[7Ø];
[R56]   14:   submenufeld=array[1..15] of boolean;
```

```
[ØØØ]   15:
[RNØ]   16: var
[KF5]   17:   hlinewidth,vlinewidth,menumode,
[8R6]   18:   aktscreen,normrahmen,hotattr,hirahmen,
[1C5]   19:   selectattribut,xmax,ymax:byte;
[ØY6]   2Ø:   hotflag,sideflag,sidemenu,activewindow,
[FT5]   21:   color,shadow,save,escflag:boolean;
[LS1]   22:   bild:word;
[536]   23:   screenfeld: array[1..5Ø] of pointer;
[ØØØ]   24:
[5B4]   25: procedure restorescreen;
[VP3]   26: procedure copyscreen;
[2H3]   27: procedure nocursor;
[XZ2]   28: procedure cursor;
[VM1]   29: procedure
[L76]   3Ø:    drawline(hpos,vpos,hpos2,xpos:byte;
[W65]   31:             attribut:word;linesign:byte);
[RA5]   32: procedure getwindpos(var x1,y1,
[SS1]   33:                      x2,y2:byte);
[E96]   34: procedure windhline(vpos,hpos:byte;
[PS2]   35:                     attribut:word);
[MB6]   36: procedure windvline(hpos,vpos:byte;
[PS2]   37:                     attribut:word);
[ZD6]   38: procedure nurrahmen(x1,y1,z1,zz:byte;
[SF2]   39:                     title:string);
[ZR5]   4Ø: procedure rahmen(x1,y1,z1,zz:byte;
[SF2]   41:                  title:string);
[8U6]   42: procedure verticalmenu(x1,y1,zahl:byte;
[4Z1]   43:                        a:menufeld);
[P67]   44: procedure horizontalmenu(ypos,zahl:byte;
[4Z1]   45:                          a:menufeld);
[VM1]   46: procedure
[Ø86]   47:     hormenu(ypos,zahl:byte; a:menufeld;
[BL2]   48:             b:submenufeld);
[WT6]   49: procedure eingabe(var a:string; var esc:
[WT3]   5Ø:                   boolean; maxlen:byte);
[VK6]   51: procedure screensave(x1,y1,x2,y2:byte);
[RW5]   52: procedure winfill(x1,y1,x2,y2:byte;
[N24]   53:                   zeichen:char;attr:byte);
[ZN3]   54: procedure decpointer;
[WZ2]   55: procedure nomaus;
[PH3]   56: procedure showmaus;
[8M5]   57: function offset(x,y:byte):word;
[L86]   58: function correct(x1,y1:byte):boolean;
[ØØØ]   59:
[4L2]   6Ø: implementation
[6XØ]   61: const
[ØW4]   62:  graf:array[1..4,1..11] of byte=
[HØ4]   63:       ((218,196,191,179,192,217,195,18Ø,
[SL1]   64:       194,193,197),
[614]   65:       (214,196,183,186,211,189,199,182,
[FH1]   66:       21Ø,2Ø8,215),
[XZ3]   67:       (213,2Ø5,184,179,212,19Ø,198,181,
[GK1]   68:       2Ø9,2Ø7,216),
[SY3]   69:       (2Ø1,2Ø5,187,186,2ØØ,188,2Ø4,185,
[CP1]   7Ø:       2Ø3,2Ø2,2Ø6));
[ØØØ]   71:
[STØ]   72: type
[EA6]   73:     screenrecord = record wmin,wmax:word;
[DK2]   74:                          x1,y1,x2,y2:byte
[ZNØ]   75:                          end;
[T65]   76:      ptrec = record ofs,seg:word end;
[ZN4]   77:      wrec = record lo,hi:byte end;
[ØØØ]   78:
[ØØØ]   79:
[B43]   8Ø: var right:boolean;
[ØØØ]   81:
[392]   82: {$l windows}
[397]   83: function offset(x,y:byte):word; external;
[X55]   84: procedure nocursor; external;
[8T4]   85: procedure cursor; external;
[ØØØ]   86:
[FP4]   87: function min(a,b:byte):byte;
[MUØ]   88: begin
[3P4]   89: if a<b then min:=a else min:=b
[ZNØ]   9Ø: end;
[ØØØ]   91:
[WZ2]   92: procedure nomaus;
[MUØ]   93: begin
[7S4]   94: if mausavail then hidecursor
[ZNØ]   95: end;
[ØØØ]   96:
[PH3]   97: procedure showmaus;
[MUØ]   98: begin
[YV4]   99: if mausavail then showcursor
[ZNØ]  1ØØ: end;
[ØØØ]  1Ø1:
[ZN3]  1Ø2: procedure decpointer;
[MUØ]  1Ø3: begin
[1R5]  1Ø4:  if aktscreen >Ø then dec(aktscreen)
[ZNØ]  1Ø5: end;
[ØØØ]  1Ø6:
[ØØØ]  1Ø7:
[ZE6]  1Ø8: procedure winshadow(x1,y1,x2,y2:byte);
[YA3]  1Ø9: var w:word; i:byte;
[MUØ]  11Ø: begin
[DC1]  111: nomaus;
[7R4]  112: for i:=y1+1 to min(y2+1,ymax) do
[MUØ]  113:     begin
[HS4]  114:     w:=memw[bild:offset(x2+1,i)];
```

```
[M32]  115:      if color then
[Z84]  116:         wrec(w).hi:=wrec(w).hi and 7
[ZC3]  117:                  else wrec(w).lo:=176;
[PF2]  118:      if x2<xmax then
[CT4]  119:          memw[bild:offset(x2+1,i)]:=w;
[HK2]  12Ø:      if x2+1<xmax then
[SN4]  121:          memw[bild:offset(x2+2,i)]:=w
[ZNØ]  122:      end;
[ØØØ]  123:
[VF2]  124: if y2<ymax then
[SE4]  125:        for i:=x1+2 to x2 do    { x1+2 }
[MUØ]  126:           begin
[GU4]  127:           w:=memw[bild:offset(i,y2+1)];
[M32]  128:           if color then
[Z84]  129:              wrec(w).hi:=wrec(w).hi and 7
[ZC3]  13Ø:                       else wrec(w).lo:=176;
[YH4]  131:           memw[bild:offset(i,y2+1)]:=w
[ZNØ]  132:           end;
[ØØØ]  133:
[NG1]  134: showmaus
[ZNØ]  135: end;
[ØØØ]  136:
[RW5]  137: procedure winfill(x1,y1,x2,y2:byte;
[664]  138:                   zeichen:char; attr:byte);
[CL3]  139: var w:word; i,j:byte;
[MUØ]  14Ø: begin
[DC1]  141: nomaus;
[8U2]  142: wrec(w).hi:=attr;
[S54]  143: wrec(w).lo:=ord(zeichen);
[LV2]  144: for i:=x1 to x2 do
[TV2]  145:       for j:=y1 to y2 do
[RC4]  146:           memw[bild:offset(i,j)]:=w;
[NG1]  147: showmaus
[ZNØ]  148: end;
[ØØØ]  149:
[ØØØ]  15Ø:
[VK6]  151: procedure screensave(x1,y1,x2,y2:byte);
[ØK6]  152:  var i,j:byte; size:word; nx2,ny2:byte;
[6R2]  153:      u:screenrecord;
[BE1]  154:      q: ^word;
[MUØ]  155:  begin
[DC1]  156:  nomaus;
[382]  157:   if shadow then
[MUØ]  158:      begin
[854]  159:      if x2<xmax-1 then nx2:=x2+2
[ZRØ]  16Ø:       else
[LW3]  161:        if x2<xmax then nx2:=x2+1
[E72]  162:          else nx2:=x2;
[YW3]  163:      if y2<ymax then ny2:=y2+1
[Ø42]  164:                 else ny2:=y2
[4MØ]  165:      end
[WP1]  166:    else begin
[B91]  167:         nx2:=x2;
[P71]  168:         ny2:=y2
[ZNØ]  169:         end;
[LL5]  17Ø:   size:=((nx2+1-x1)*(ny2+1-y1)) shl 1 +8;
[EM2]  171:   inc(aktscreen);
[2R5]  172:   u.wmin:=windmin; u.wmax:=windmax;
[AT4]  173:   u.x1:=x1; u.y1:=y1; u.x2:=nx2;
[VM1]  174:   u.y2:=ny2;
[P46]  175:   getmem(screenfeld[aktscreen],size);
[5C4]  176:   q:=screenfeld[aktscreen];
[L12]  177:   move(u,q^,8);
[R63]  178:   inc(ptrec(q).ofs,8);
[R63]  179:   for i:=y1 to ny2 do
[M63]  18Ø:         for j:=x1 to nx2 do
[MUØ]  181:           begin
[2R4]  182:           q^:=memw[bild:offset(j,i)];
[A43]  183:           inc(ptrec(q).ofs,2)
[ZNØ]  184:           end;
[NG1]  185:   showmaus
[ZNØ]  186: end;
[ØØØ]  187:
[5B4]  188: procedure restorescreen;
[3U5]  189:  var i,j,x1,y1,x2,y2:byte; q: ^word;
[6R2]  19Ø:      u:screenrecord;
[MUØ]  191:  begin
[DC1]  192:  nomaus;
[5C4]  193:  q:=screenfeld[aktscreen];
[S52]  194:  move(q^,u,8);
[R63]  195:  inc(ptrec(q).ofs,8);
[KR5]  196:  windmin:=u.wmin; windmax:=u.wmax;
[M86]  197:  x1:=u.x1; x2:=u.x2; y1:=u.y1; y2:=u.y2;
[ØØØ]  198:
[RV2]  199:   for i:=y1 to y2 do
[NV2]  2ØØ:        for j:=x1 to x2 do
[MUØ]  2Ø1:           begin
[2R4]  2Ø2:           memw[bild:offset(j,i)]:=q^;
[A43]  2Ø3:           inc(ptrec(q).ofs,2)
[ZNØ]  2Ø4:           end;
[ØØØ]  2Ø5:
[KD5]  2Ø6:   freemem(screenfeld[aktscreen],
[K34]  2Ø7:           ((y2+1-y1)*(x2+1-x1)) shl 1 +8);
[9L2]  2Ø8:   dec(aktscreen);
[NG1]  2Ø9:   showmaus
[ZNØ]  21Ø: end;
[ØØØ]  211:
[VP3]  212: procedure copyscreen;
[4V6]  213:  var i,j,x1,y1,x2,y2:byte; u:screenrecord;
[YF1]  214:      q: word;
[MUØ]  215:  begin
```

```
[DC1]  216:   nomaus;
[5C4]  217:   q:=screenfeld[aktscreen];
[S52]  218:   move(q^,u,8);
[M86]  219:   x1:=u.x1; x2:=u.x2; y1:=u.y1; y2:=u.y2;
[KR5]  22Ø:   windmin:=u.wmin; windmax:=u.wmax;
[R63]  221:   inc(ptrec(q).ofs,8);
[ØØØ]  222:
[RV2]  223:     for i:=y1 to y2 do
[NV2]  224:        for j:=x1 to x2 do
[MUØ]  225:           begin
[2R4]  226:           memw[bild:offset(j,i)]:=q^;
[A43]  227:           inc(ptrec(q).ofs,2)
[ZNØ]  228:           end;
[NG1]  229:     showmaus
[ZNØ]  23Ø: end;
[ØØØ]  231:
[VM1]  232: procedure
[L76]  233:    drawline(hpos,vpos,hpos2,xpos:byte;
[W65]  234:             attribut:word;linesign:byte);
[4Ø4]  235:  var x1,y1,x2,y2,i,j:byte;
[MUØ]  236:  begin
[DC1]  237:  nomaus;
[264]  238:  attribut:=attribut shl 8;
[ZV5]  239:  j:=((hlinewidth-1) shl 1)+vlinewidth;
[SG5]  24Ø:  memw[bild:offset(hpos,vpos-1)]:=
[KB3]  241:                graf[j,7]+attribut;
[NR6]  242:  memw[bild:offset(hpos+hpos2+1,vpos-1)]:=
[LB3]  243:                graf[j,8]+attribut;
[VX2]  244:  if linesign=Ø then
[H45]  245:    for i:=hpos+1 to (hpos+hpos2) do
[1Y4]  246:       memw[bild:offset(i,vpos-1)]:=
[143]  247:                graf[j,2]+attribut
[ZRØ]  248:  else
[H45]  249:    for i:=hpos+1 to (hpos+hpos2) do
[1Y4]  25Ø:       memw[bild:offset(i,vpos-1)]:=
[D63]  251:                 linesign+attribut;
[Z82]  252:  if xpos>Ø then
[CA6]  253:     memw[bild:offset(hpos+xpos-1,vpos-1)]
[1N3]  254:              :=graf[j,11]+attribut;
[NG1]  255:  showmaus
[ZNØ]  256:  end;
[ØØØ]  257:
[RA5]  258: procedure getwindpos(var x1,y1,
[SS1]  259:                       x2,y2:byte);
[MUØ]  26Ø: begin
[YH5]  261:  x1:=lo(windmin)+1; y1:=hi(windmin)+1;
[XK5]  262:  x2:=lo(windmax)+1; y2:=hi(windmax)+1;
[ZNØ]  263: end;
[ØØØ]  264:
[ØØØ]  265:
[E96]  266: procedure windhline(vpos,hpos:byte;
[PS2]  267:                     attribut:word);
[4Ø4]  268:  var x1,y1,x2,y2,i,j:byte;
[MUØ]  269:  begin
[DC1]  27Ø:  nomaus;
[264]  271:  attribut:=attribut shl 8;
[ZV5]  272:  j:=((hlinewidth-1) shl 1)+vlinewidth;
[5V3]  273:  getwindpos(x1,y1,x2,y2);
[6X5]  274:  memw[bild:offset(x1-1,y1+vpos-1)]:=
[KB3]  275:           graf[j,7]+attribut;
[8X5]  276:  memw[bild:offset(x2+1,y1+vpos-1)]:=
[LB3]  277:           graf[j,8]+attribut;
[LV2]  278:  for i:=x1 to x2 do
[EA5]  279:        memw[bild:offset(i,y1+vpos-1)]:=
[DB3]  28Ø:              graf[j,2]+attribut;
[F82]  281:  if hpos>Ø then
[1A6]  282:     memw[bild:offset(x1+hpos-1,y1+vpos-1)]
[1N3]  283:         :=graf[j,11]+attribut;
[NG1]  284:  showmaus
[ZNØ]  285:  end;
[ØØØ]  286:
[MB6]  287: procedure windvline(hpos,vpos:byte;
[PS2]  288:                     attribut:word);
[4Ø4]  289:  var x1,y1,x2,y2,i,j:byte;
[MUØ]  29Ø:  begin
[DC1]  291:  nomaus;
[264]  292:  attribut:=attribut shl 8;
[ZV5]  293:  j:=((hlinewidth-1) shl 1)+vlinewidth;
[5V3]  294:  getwindpos(x1,y1,x2,y2);
[ZS5]  295:  memw[bild:offset(x1+vpos-1,y1-1)]:=
[MB3]  296:           graf[j,9]+attribut;
[VS5]  297:  memw[bild:offset(x1+vpos-1,y2+1)]:=
[AF3]  298:           graf[j,1Ø]+attribut;
[RV2]  299:  for i:=y1 to y2 do
[BB5]  3ØØ:       memw[bild:offset(x1+vpos-1,i)]:=
[FB3]  3Ø1:            graf[j,4]+attribut;
[F82]  3Ø2:  if hpos>Ø then
[596]  3Ø3:     memw[bild:offset(x1+vpos-1,y1+hpos-1)]
[1N3]  3Ø4:              :=graf[j,11]+attribut;
[NG1]  3Ø5:  showmaus
[ZNØ]  3Ø6:  end;
[ØØØ]  3Ø7:
[AK3]  3Ø8: procedure nurrahmen;
[4Y5]  3Ø9:  var i,j,k,u,x2,y2:byte; attr,z:word;
[MUØ]  31Ø:  begin
[DC1]  311:  nomaus;
[2N3]  312:  x2:=x1+zl+1; y2:=y1+zz+1;
[CØ6]  313:  k:=length(title)+2; if k>zl then k:=Ø;
```

Listing 12. »windows.pas« ruft die Assembler-Treiberprogramme zur Menü- und Fensterbehandlung auf

```
[UW5]  314:  if save then screensave(x1,y1,x2,y2);
[XZ6]  315:  if activewindow then attr:=hirahmen shl 8
[AP4]  316:       else attr:=normrahmen shl 8;
[ZV5]  317:  j:=((hlinewidth-1) shl 1)+vlinewidth;
[573]  318:  for i:=x1+1 to x2-1 do
[MUØ]  319:   begin
[WW6]  32Ø:   memw[bild:offset(i,y1)]:=graf[j,2]+attr;
[YW6]  321:   memw[bild:offset(i,y2)]:=graf[j,2]+attr;
[ZNØ]  322:   end;
[CN1]  323:  if k>2 then
[MUØ]  324:    begin
[KØ3]  325:    title:=' '+title+' ';
[MA2]  326:    u:=(zl-k) shr 1;
[Y71]  327:    u:=u+x1;
[LH2]  328:    for i:=1 to k do
[XK4]  329:      memw[bild:offset(i+u,y1)]:=
[GS4]  33Ø:           ord(title[i])+textattr shl 8
[ZNØ]  331:    end;
[773]  332:  for i:=y1+1 to y2-1 do
[MUØ]  333:   begin
[4Z6]  334:   memw[bild:offset(x1,i)]:=graf[j,4]+attr;
[8Z6]  335:   memw[bild:offset(x2,i)]:=graf[j,4]+attr;
[ZNØ]  336:   end;
[ØZ6]  337:  memw[bild:offset(x2,y1)]:=graf[j,3]+attr;
[CZ6]  338:  memw[bild:offset(x2,y2)]:=graf[j,6]+attr;
[PY6]  339:  memw[bild:offset(x1,y1)]:=graf[j,1]+attr;
[5Z6]  34Ø:  memw[bild:offset(x1,y2)]:=graf[j,5]+attr;
[1Z5]  341:  if shadow then winshadow(x1,y1,x2,y2);
[NG1]  342:  showmaus
[ZNØ]  343:  end;
[ØØØ]  344:
[ZR5]  345: procedure rahmen(x1,y1,zl,zz:byte;
[SF2]  346:                   title:string);
[4Y5]  347:  var i,j,k,u,x2,y2:byte; attr,z:word;
[MUØ]  348:  begin
[DC1]  349:  nomaus;
[2N3]  35Ø:  x2:=x1+zl+1; y2:=y1+zz+1;
[CØ6]  351:  k:=length(title)+2; if k>zl then k:=Ø;
[UW5]  352:  if save then screensave(x1,y1,x2,y2);
[XZ6]  353:  if activewindow then attr:=hirahmen shl 8
[AP4]  354:               else attr:=normrahmen shl 8;
[ZV5]  355:  j:=((hlinewidth-1) shl 1)+vlinewidth;
[573]  356:  for i:=x1+1 to x2-1 do
[MUØ]  357:   begin
[WW6]  358:   memw[bild:offset(i,y1)]:=graf[j,2]+attr;
[YW6]  359:   memw[bild:offset(i,y2)]:=graf[j,2]+attr;
[ZNØ]  36Ø:   end;
[CN1]  361:  if k>2 then
[MUØ]  362:    begin
[KØ3]  363:    title:=' '+title+' ';
[MA2]  364:    u:=(zl-k) shr 1;
[Y71]  365:    u:=u+x1;
[LH2]  366:    for i:=1 to k do
[XK4]  367:       memw[bild:offset(i+u,y1)]:=
[GS4]  368:            ord(title[i])+textattr shl 8
[ZNØ]  369:    end;
[773]  37Ø:  for i:=y1+1 to y2-1 do
[MUØ]  371:   begin
[4Z6]  372:   memw[bild:offset(x1,i)]:=graf[j,4]+attr;
[8Z6]  373:   memw[bild:offset(x2,i)]:=graf[j,4]+attr;
[ZNØ]  374:   end;
[ØZ6]  375:  memw[bild:offset(x2,y1)]:=graf[j,3]+attr;
[CZ6]  376:  memw[bild:offset(x2,y2)]:=graf[j,6]+attr;
[PY6]  377:  memw[bild:offset(x1,y1)]:=graf[j,1]+attr;
[5Z6]  378:  memw[bild:offset(x1,y2)]:=graf[j,5]+attr;
[1Z5]  379:  if shadow then winshadow(x1,y1,x2,y2);
[K14]  38Ø:  window(x1+1,y1+1,x2-1,y2-1);
[NG1]  381:  showmaus
[ZNØ]  382:  end;
[ØØØ]  383:
[ØØØ]  384:
[L86]  385: function correct(x1,y1:byte):boolean;
[7L6]  386:  var ch:string; esc,s:boolean; attr:byte;
[MUØ]  387:  begin
[EB1]  388:   s:=save;
[S12]  389:   save:=true;
[S23]  39Ø:   rahmen(x1,y1,35,1,'');
[ER6]  391:   attr:=textattr; textattr:=normrahmen;
[DB1]  392:   clrscr;
[6XØ]  393:   write
[475]  394:   (' Ist die Eingabe korrekt ? (j/n)');
[ES2]  395:   eingabe(ch,esc,1);
[MG5]  396:   if (ord(ch[1]) in [1Ø6,74]) or (ch='')
[K43]  397:         then correct:=true
[DE3]  398:         else correct:=false;
[AG4]  399:   if esc then correct:=false;
[SU2]  4ØØ:   textattr:=attr;
[GH2]  4Ø1:   restorescreen;
[XB1]  4Ø2:   save:=s
[ZNØ]  4Ø3: end;
[ØØØ]  4Ø4:
[VM1]  4Ø5: procedure
[PC6]  4Ø6:   eingabe(var a:string; var esc:boolean;
[882]  4Ø7:           maxlen:byte);
[CA3]  4Ø8: var ch,relpos:byte;
[ØØØ]  4Ø9:
[RM2]  41Ø:     procedure back;
[MUØ]  411:     begin
[KR2]  412:     if relpos<>Ø then
[MUØ]  413:       begin
[Z14]  414:       gotoxy(wherex-1,wherey);
[EL1]  415:       write(' ');
[324]  416:       gotoxy(wherex-2,wherey);
[MW2]  417:       relpos:=relpos-1;
[BZ2]  418:       if relpos=maxlen-2
[FA4]  419:          then delete(a,length(a)-1,2)
[H34]  42Ø:          else delete(a,length(a),1)
[4MØ]  421:       end
[ZNØ]  422:     end;
[ØØØ]  423:
[MUØ]  424: begin
[ØD1]  425: cursor;
[AUØ]  426: a:='';
[ØP1]  427: relpos:=Ø;
[KV2]  428: ch:=ord(readkey);
[UZ3]  429: while (ch<>27)and(ch<>13) do
[MUØ]  43Ø:   begin
[GT1]  431:   if ch=Ø then
[MUØ]  432:     begin
[KV2]  433:     ch:=ord(readkey);
[PH3]  434:     if ch=75 then back else
[YX1]  435:      if ch=83 then
[MUØ]  436:       begin
[FM1]  437:       write(' ');
[Z14]  438:       gotoxy(wherex-1,wherey);
[7X3]  439:       if length(a)=maxlen then
[463]  44Ø:              delete(a,length(a),1)
[4MØ]  441:       end
[4MØ]  442:     end
[ZRØ]  443:    else
[LH2]  444:     if ch=8 then back
[ZRØ]  445:       else
[MUØ]  446:        begin
[DC1]  447:        nomaus;
[VC2]  448:        write(chr(ch));
[BM1]  449:        showmaus;
[LU3]  45Ø:        if relpos=maxlen-1 then
[MUØ]  451:           begin
[Z14]  452:           gotoxy(wherex-1,wherey);
[7X3]  453:           if length(a)=maxlen then
[NB3]  454:                delete(a,length(a),1);
[SU1]  455:           a:=a+chr(ch)
[4MØ]  456:           end
[WP1]  457:        else begin
[EY1]  458:             a:=a+chr(ch);
[PS2]  459:             relpos:=relpos+1
[4MØ]  46Ø:             end
[ZNØ]  461:        end;
[WR2]  462:   ch:=ord(readkey)
[ZNØ]  463:   end;
[5E6]  464: if ch=27 then esc:=true else esc:=false;
[ZNØ]  465: end;
[ØØØ]  466:
[ØØØ]  467:
[8W6]  468: procedure verticalmenu(x1,y1, zahl:byte;
[4Z1]  469:                         a:menufeld);
[4E5]  47Ø: var i,l,zl,ch,pos,a1,b1,a2,b2:byte;
[G84]  471:     func,fertig,flag:boolean;
[UN4]  472:     hotpos:array[1..2Ø] of byte;
[5Z4]  473:     hotchar: array[1..2Ø] of char;
[R62]  474:     s:string[4Ø];
[ØØØ]  475:
[TZ2]  476:  procedure invers;
[AS2]  477:  var i,attr:byte;
[MUØ]  478:  begin
[WG2]  479:   flag:=not flag;
[VH5]  48Ø:   if flag then attr:=selectattribut
[AM3]  481:           else attr:=textattr;
[1P3]  482:   for i := x1+1 to x1+zl do
[TY5]  483:     mem[bild:offset(i,pos+y1)+1]:=attr;
[HK4]  484:   if not flag and hotflag then
[AM5]  485:      mem[bild:offset(x1+hotpos[pos],
[GD3]  486:                pos+y1)+1]:=hotattr
[ZNØ]  487:   end;
[ØØØ]  488:
[MUØ]  489:  begin
[GYØ]  49Ø:  zl:=1;
[ZL3]  491:  getwindpos(a1,b1,a2,b2);
[983]  492:  for i:=1 to zahl do
[YW2]  493:       if length(a[i])>zl
[3U3]  494:            then zl:=length(a[i]);
[736]  495:  if hotflag then inc(zl,1) else inc(zl,2);
[DY2]  496:  if menumode=Ø then
[MUØ]  497:    begin
[8V3]  498:    rahmen(x1,y1,zl,zahl,'');
[C81]  499:    pos:=1;
[3D1]  5ØØ:    end else
[MUØ]  5Ø1:       begin
[2H2]  5Ø2:       pos:=menumode;
[9T4]  5Ø3:       window(x1+1,y1+1,x1+zl,y1+zahl)
[ZNØ]  5Ø4:       end;
[DC1]  5Ø5: nomaus;
[DB1]  5Ø6: clrscr;
[983]  5Ø7: for i:=1 to zahl do
[CF2]  5Ø8:    if hotflag then
[MUØ]  5Ø9:       begin
[ZN3]  51Ø:       s:=scanleft(a[i],'^');
[A24]  511:       hotpos[i]:=length(s)+2;
[TX1]  512:       gotoxy(2,i);
[N34]  513:       s:=s+scanright(a[i],'^');
[E96]  514:       hotchar[i]:=upchar(s[hotpos[i]-1]);
[9G1]  515:       write(s);
```

```
[735]  516:        mem[bild:offset(x1+hotpos[i],
[9R2]  517:                 i+y1)+1]:=hotattr
[4MØ]  518:        end
[WP1]  519:       else begin
[TX1]  52Ø:            gotoxy(2,i);
[AY1]  521:            write(a[i])
[ZNØ]  522:            end;
[GX2]  523: window(a1,b1,a2,b2);
[E54]  524: flag:=false; func:=false;
[2D1]  525: invers;
[BM1]  526: showmaus;
[ØØØ]  527:
[TV5]  528:  ch:=1; escflag:=false; fertig:=false;
[Z83]  529:  while not fertig do
[MUØ]  53Ø:    begin
[ØT5]  531:    if keypressed then ch:=ord(readkey);
[GT1]  532:    if ch=Ø then
[MUØ]  533:       begin
[KV2]  534:       ch:=ord(readkey);
[TW3]  535:       func:=ch in [8Ø,72,75,77];
[GM3]  536:       if not func then ch:=1;
[ZNØ]  537:       end;
[UB4]  538:    if hotflag and not func then
[983]  539:    for i:=1 to zahl do
[DH5]  54Ø:     if upchar(chr(ch))=hotchar[i] then
[MUØ]  541:        begin
[DC1]  542:        nomaus;
[2D1]  543:        invers;
[MA1]  544:        pos:=i;
[2D1]  545:        invers;
[BM1]  546:        showmaus;
[YVØ]  547:        ch:=13
[ZNØ]  548:        end;
[YG3]  549:    if (ch=72)or(ch=8Ø) then
[CY1]  55Ø:       if func then
[MUØ]  551:          begin
[DC1]  552:          nomaus;
[2D1]  553:          invers;
[WN3]  554:          if ch=72 then pos:=pos-1
[YM2]  555:                   else pos:=pos+1;
[2W3]  556:          if pos=Ø then pos:=zahl;
[V14]  557:          if pos=zahl+1 then pos:=1;
[CH3]  558:          invers; func:=false;
[BM1]  559:          showmaus;
[PWØ]  56Ø:          ch:=1;
[ZNØ]  561:          end;
[5G2]  562:     if button1 then
[XM2]  563:      if aktx>=x1 then
[ØX2]  564:       if aktx<x1+zl then
[4N2]  565:        if akty>=y1 then
[CH2]  566:         if akty<y1+zahl
[ZN1]  567:           then begin
[DC1]  568:                nomaus;
[2D1]  569:                invers;
[LL2]  57Ø:                pos:=akty-y1+1;
[2D1]  571:                invers;
[BM1]  572:                showmaus;
[A31]  573:                ch:=13;
[T54]  574:                repeat until not button1
[ZNØ]  575:                end;
[GZ2]  576:     if ch=13 then begin
[3F2]  577:                  menumode:=pos;
[U52]  578:                  fertig:=true
[ZNØ]  579:                  end;
[RU3]  58Ø:     if button2 or (ch=27) then
[MUØ]  581:           begin
[ME2]  582:           escflag:=true;
[G92]  583:           fertig:=true;
[NR2]  584:           sideflag:=false;
[8G2]  585:           if button2 then
[U54]  586:              repeat until not button2
[ZNØ]  587:           end;
[RH3]  588:     if (ch=75)or(ch=77) then
[CY1]  589:       if func then
[ØP2]  59Ø:         if sidemenu then
[MUØ]  591:           begin
[ME2]  592:           escflag:=true;
[G92]  593:           fertig:=true;
[3P2]  594:           sideflag:=true;
[362]  595:           right:=(ch=75)
[4MØ]  596:           end
[ZNØ]  597:    end;
[NF3]  598:  if activewindow then
[MUØ]  599:    begin
[3L3]  6ØØ:    activewindow:=false;
[EX2]  6Ø1:    if save then begin
[A32]  6Ø2:                 save:=false;
[8V3]  6Ø3:                 rahmen(x1,y1,zl,zahl,'');
[CU1]  6Ø4:                 save:=true
[4MØ]  6Ø5:                 end
[4P4]  6Ø6:            else rahmen(x1,y1,zl,zahl,'');
[5A3]  6Ø7:    activewindow:=true
[4MØ]  6Ø8:    end
[ZNØ]  6Ø9:  end;
[ØØØ]  61Ø:
[P67]  611: procedure horizontalmenu(ypos,zahl:byte;
[4Z1]  612:                          a:menufeld);
[X46]  613: var i,j,l,x1,x2,attr,platz,ch,pos,wx,
[EN5]  614:     v,vert:byte; func,fertig:boolean;
[R35]  615:     spos,epos:array[1..2Ø] of byte;
[W74]  616:     s:string[4Ø]; t:menufeld;
[UN4]  617:     hotpos:array[1..2Ø] of byte;
```

```
[TT4]  618:     hotchar:array[1..2Ø] of char;
[ØØØ]  619:
[TZ2]  62Ø:  procedure invers;
[4H2]  621:  var attr:byte;
[MUØ]  622:  begin
[EV2]  623:  attr:=textattr;
[4L4]  624:  textattr:=selectattribut;
[H94]  625:  gotoxy(spos[pos],ypos);
[NA5]  626:  write(t[pos]); textattr:=attr;
[ZNØ]  627:  end;
[ØØØ]  628:
[MUØ]  629: begin
[5C5]  63Ø:  x1:=lo(windmin); x2:=lo(windmax);
[9P4]  631:  v:=hi(windmin); platz:=x2-x1;
[283]  632:  j:=Ø; vert:=v+ypos;
[3G6]  633:  for i:=1 to zahl do j:=j+length(a[i])-1;
[US2]  634:  platz:=platz-j-1;
[VB4]  635:  if (platz>Ø)and(zahl>1) then
[MUØ]  636:     begin
[EE3]  637:     j:=platz div (zahl-1);
[V65]  638:     wx:=2+((platz mod (zahl-1)) div 2);
[983]  639:     for i:=1 to zahl do
[MUØ]  64Ø:       begin
[YB2]  641:       spos[i]:=wx;
[ØX4]  642:       epos[i]:=wx+length(a[i])-1;
[DT3]  643:       inc(wx,length(a[i])+j-1)
[4MØ]  644:       end
[ZNØ]  645:     end;
[ØØØ]  646:
[183]  647:  spos[zahl+1]:=x2+1;
[VX3]  648:  if menumode=Ø then pos:=1
[3E3]  649:                else pos:=menumode;
[DC1]  65Ø:  nomaus;
[983]  651:  for i:=1 to zahl do
[CF2]  652:    if hotflag then
[MUØ]  653:      begin
[ZN3]  654:      s:=scanleft(a[i],'^');
[8R3]  655:      hotpos[i]:=length(s);
[6T3]  656:      gotoxy(spos[i],ypos);
[N34]  657:      s:=s+scanright(a[i],'^');
[696]  658:      hotchar[i]:=upchar(s[hotpos[i]+1]);
[T13]  659:      write(s); t[i]:=s;
[3K6]  66Ø:      mem[bild:offset(x1+spos[i]+hotpos[i],
[SU2]  661:              vert)+1]:=hotattr
[4MØ]  662:      end
[WP1]  663:     else begin
[6T3]  664:          gotoxy(spos[i],ypos);
[S44]  665:          write(a[i]); t[i]:=a[i]
[ZNØ]  666:          end;
[2D1]  667:  invers;
[BM1]  668:  showmaus;
[ØØØ]  669:
[TV5]  67Ø:   ch:=1; escflag:=false; fertig:=false;
[A32]  671:   func:=false;
[Z83]  672:   while not fertig do
[MUØ]  673:      begin
[ØT5]  674:      if keypressed then ch:=ord(readkey);
[GT1]  675:      if ch=Ø then
[MUØ]  676:         begin
[KV2]  677:         ch:=ord(readkey);
[N83]  678:         func:=ch in [75,77];
[4D3]  679:         if not func then ch:=1
[ZNØ]  68Ø:         end;
[UB4]  681:      if hotflag and not func then
[983]  682:        for i:=1 to zahl do
[DH5]  683:         if upchar(chr(ch))=hotchar[i] then
[MUØ]  684:            begin
[DC1]  685:            nomaus;
[H94]  686:            gotoxy(spos[pos],ypos);
[2G2]  687:            write(t[pos]);
[CF2]  688:            if hotflag then
[T35]  689:            mem[bild:offset(x1+spos[pos]+
[GD5]  69Ø:             hotpos[pos],vert)+1]:=hotattr;
[MA1]  691:            pos:=i;
[2D1]  692:            invers;
[BM1]  693:            showmaus;
[YVØ]  694:            ch:=13
[ZNØ]  695:            end;
[UC5]  696:      if (ch=75)or(ch=77)or sideflag then
[2W3]  697:        if func or sideflag then
[MUØ]  698:           begin
[DC1]  699:           nomaus;
[H94]  7ØØ:           gotoxy(spos[pos],ypos);
[2G2]  7Ø1:           write(t[pos]);
[CF2]  7Ø2:           if hotflag then
[T35]  7Ø3:           mem[bild:offset(x1+spos[pos]+
[GD5]  7Ø4:            hotpos[pos],vert)+1]:=hotattr;
[TW4]  7Ø5:           if (ch=75)or(right and sideflag)
[CK2]  7Ø6:                    then pos:=pos-1
[YM2]  7Ø7:                    else pos:=pos+1;
[2W3]  7Ø8:           if pos=Ø then pos:=zahl;
[V14]  7Ø9:           if pos=zahl+1 then pos:=1;
[2D1]  71Ø:           invers;
[FR3]  711:           showmaus; func:=false;
[NR2]  712:           sideflag:=false;
[5RØ]  713:           ch:=1
[ZNØ]  714:           end;
[ØØØ]  715:
[5G2]  716:      if button1 then
```

Listing 12. »windows.pas« ruft die Assembler-Treiberprogramme zur Menü- und Fensterbehandlung auf

```
[ND3]  717:        if akty=ypos-1+v then
[3G2]  718:         if aktx>x1 then
[VN2]  719:          if aktx<x2-1 then
[MUØ]  72Ø:             begin
[DC1]  721:             nomaus;
[H94]  722:             gotoxy(spos[pos],ypos);
[2G2]  723:             write(t[pos]);
[CF2]  724:             if hotflag then
[T35]  725:             mem[bild:offset(x1+spos[pos]+
[GD5]  726:              hotpos[pos],vert)+1]:=hotattr;
[EZØ]  727:             wx:=1;
[RF4]  728:             while aktx+2>spos[wx+1]+x1
[HN1]  729:                   do inc(wx);
[FD4]  73Ø:             if aktx+1<epos[wx]+x1 then
[MUØ]  731:               begin
[DH1]  732:               pos:=wx;
[YVØ]  733:               ch:=13
[ZNØ]  734:               end;
[2D1]  735:             invers;
[BM1]  736:             showmaus;
[T54]  737:             repeat until not button1
[ZNØ]  738:             end;
[GZ2]  739:      if ch=13 then begin
[3F2]  74Ø:                   menumode:=pos;
[U52]  741:                   fertig:=true
[ZNØ]  742:                   end;
[RU3]  743:      if button2 or (ch=27) then
[MUØ]  744:         begin
[ME2]  745:         escflag:=true;
[G92]  746:         fertig:=true;
[XM1]  747:         if button2
[WY4]  748:            then repeat until not button2
[4MØ]  749:         end
[4MØ]  75Ø:     end
[ZNØ]  751: end;
[ØØØ]  752:
[VM1]  753: procedure
[Ø86]  754:    hormenu(ypos,zahl:byte; a:menufeld;
[BL2]  755:            b:submenufeld);
[FK6]  756: var i,j,l,x1,x2,attr,platz,ch,pos,wx,v,
[PD5]  757:     vert:byte; func,fertig:boolean;
[R35]  758:     spos,epos:array[1..2Ø] of byte;
[W74]  759:     s:string[4Ø]; t:menufeld;
[UN4]  76Ø:     hotpos:array[1..2Ø] of byte;
[TT4]  761:     hotchar:array[1..2Ø] of char;
[ØØØ]  762:
[TZ2]  763:  procedure invers;
[4H2]  764:  var attr:byte;
[MUØ]  765:  begin
[EV2]  766:  attr:=textattr;
[4L4]  767:  textattr:=selectattribut;
[EU6]  768:  gotoxy(spos[pos],ypos);write(t[pos]);
[SU2]  769:  textattr:=attr;
[ZNØ]  77Ø:  end;
[ØØØ]  771:
[MUØ]  772: begin
[5C5]  773:  x1:=lo(windmin); x2:=lo(windmax);
[DH2]  774:  v:=hi(windmin);
[ØE5]  775:  platz:=x2-x1; j:=Ø; vert:=v+ypos;
[3G6]  776:  for i:=1 to zahl do j:=j+length(a[i])-1;
[US2]  777:  platz:=platz-j-1;
[VB4]  778:  if (platz>Ø)and(zahl>1) then
[MUØ]  779:    begin
[EE3]  78Ø:    j:=platz div (zahl-1);
[V65]  781:    wx:=2+((platz mod (zahl-1)) div 2);
[983]  782:       for i:=1 to zahl do
[MUØ]  783:            begin
[YB2]  784:            spos[i]:=wx;
[ØX4]  785:            epos[i]:=wx+length(a[i])-1;
[DT3]  786:            inc(wx,length(a[i])+j-1)
[4MØ]  787:            end
[ZNØ]  788:    end;
[183]  789:  spos[zahl+1]:=x2+1;
[VX3]  79Ø:  if menumode=Ø then pos:=1
[3E3]  791:               else pos:=menumode;
[DC1]  792:  nomaus;
[983]  793:  for i:=1 to zahl do
[CF2]  794:    if hotflag then
[MUØ]  795:       begin
[ZN3]  796:       s:=scanleft(a[i],'^');
[8R3]  797:       hotpos[i]:=length(s);
[6T3]  798:       gotoxy(spos[i],ypos);
[N34]  799:       s:=s+scanright(a[i],'^');
[696]  8ØØ:       hotchar[i]:=upchar(s[hotpos[i]+1]);
[T13]  8Ø1:       write(s); t[i]:=s;
[2U4]  8Ø2:       mem[bild:offset(x1+spos[i]+
[B25]  8Ø3:             hotpos[i],vert)+1]:=hotattr
[4MØ]  8Ø4:       end
[WP1]  8Ø5:     else begin
[6T3]  8Ø6:          gotoxy(spos[i],ypos);
[S44]  8Ø7:          write(a[i]); t[i]:=a[i]
[ZNØ]  8Ø8:          end;
[ØØØ]  8Ø9:
[2D1]  81Ø:  invers;
[BM1]  811:  showmaus;
[TV5]  812:  ch:=1; escflag:=false; fertig:=false;
[A32]  813:  func:=false;
[Z83]  814:  while not fertig do
[MUØ]  815:     begin
[ØT5]  816:     if keypressed then ch:=ord(readkey);
[GT1]  817:     if ch=Ø then
```

```
[MUØ]  818:        begin
[KV2]  819:        ch:=ord(readkey);
[N83]  82Ø:        func:=ch in [75,77];
[4D3]  821:        if not func then ch:=1
[ZNØ]  822:        end;
[UB4]  823:     if hotflag and not func then
[983]  824:       for i:=1 to zahl do
[DH5]  825:         if upchar(chr(ch))=hotchar[i] then
[MUØ]  826:            begin
[DC1]  827:            nomaus;
[H94]  828:            gotoxy(spos[pos],ypos);
[2G2]  829:            write(t[pos]);
[CF2]  83Ø:            if hotflag then
[T35]  831:             mem[bild:offset(x1+spos[pos]+
[GD5]  832:             hotpos[pos],vert)+1]:=hotattr;
[MA1]  833:            pos:=i;
[2D1]  834:            invers;
[BM1]  835:            showmaus;
[YVØ]  836:            ch:=13
[ZNØ]  837:            end;
[UC5]  838:     if (ch=75)or(ch=77)or sideflag then
[2W3]  839:       if func or sideflag then
[MUØ]  84Ø:          begin
[DC1]  841:          nomaus;
[H94]  842:          gotoxy(spos[pos],ypos);
[2G2]  843:          write(t[pos]);
[CF2]  844:          if hotflag then
[T35]  845:           mem[bild:offset(x1+spos[pos]+
[GD5]  846:           hotpos[pos],vert)+1]:=hotattr;
[TW4]  847:          if (ch=75)or(right and sideflag)
[CK2]  848:                then pos:=pos-1
[YM2]  849:                else pos:=pos+1;
[2W3]  85Ø:          if pos=Ø then pos:=zahl;
[V14]  851:          if pos=zahl+1 then pos:=1;
[D23]  852:          invers; showmaus;
[PK3]  853:          if b[pos] and sideflag
[TM3]  854:             then ch:=13 else ch:=1;
[ES4]  855:          sideflag:=false; func:=false
[ZNØ]  856:          end;
[5G2]  857:     if button1 then
[ND3]  858:      if akty=ypos-1+v then
[3G2]  859:       if aktx>x1 then
[VN2]  86Ø:        if aktx<x2-1 then
[MUØ]  861:          begin
[DC1]  862:          nomaus;
[H94]  863:          gotoxy(spos[pos],ypos);
[2G2]  864:          write(t[pos]);
[CF2]  865:          if hotflag then
[T35]  866:           mem[bild:offset(x1+spos[pos]+
[GD5]  867:           hotpos[pos],vert)+1]:=hotattr;
[EZØ]  868:          wx:=1;
[TU4]  869:          while aktx+2>spos[wx+1]+x1 do
[5A1]  87Ø:              inc(wx);
[FD4]  871:          if aktx+1<epos[wx]+x1 then
[MUØ]  872:             begin
[DH1]  873:             pos:=wx;
[YVØ]  874:             ch:=13
[ZNØ]  875:             end;
[2D1]  876:          invers;
[BM1]  877:          showmaus;
[T54]  878:          repeat until not button1
[ZNØ]  879:          end;
[GZ2]  88Ø:     if ch=13 then begin
[3F2]  881:                   menumode:=pos;
[U52]  882:                   fertig:=true
[ZNØ]  883:                   end;
[RU3]  884:     if button2 or (ch=27) then
[MUØ]  885:        begin
[ME2]  886:        escflag:=true;
[G92]  887:        fertig:=true;
[8G2]  888:        if button2 then
[U54]  889:           repeat until not button2
[4MØ]  89Ø:        end
[4MØ]  891:    end
[ZNØ]  892: end;
[ØØØ]  893:
[MUØ]  894: begin
[6Z5]  895: activewindow:=true; hlinewidth:=1;
[A75]  896: vlinewidth:=1; sideflag:=false;
[A76]  897: nocursor; shadow:=true; save:=true;
[B35]  898: xmax:=8Ø; ymax:=25; menumode:=Ø;
[ØX4]  899: hotflag:=false; aktscreen:=Ø;
[GT2]  9ØØ: sidemenu:=false;
[E43]  9Ø1: if lastmode<>7 then
[MUØ]  9Ø2:     begin
[BP1]  9Ø3:     bild:=$b8ØØ;
[P55]  9Ø4:     normrahmen:=blue*16+lightgray;
[UU4]  9Ø5:     selectattribut:=red*16+black;
[HK3]  9Ø6:     hirahmen:=blue*16+red;
[Y32]  9Ø7:     color:=true;
[4MØ]  9Ø8:     end
[WP1]  9Ø9:  else begin
[VN1]  91Ø:       bild:=$bØØØ;
[4D2]  911:       normrahmen:=4;
[VB3]  912:       selectattribut:=$7Ø;
[732]  913:       hirahmen:=15;
[N32]  914:       color:=false
[4MØ]  915:       end
[KNØ]  916: end.
```

Listing 12. »windows.pas« ruft die Assembler-Treiberprogramme zur Menü- und Fensterbehandlung auf

```
[XE5]     1: 8Ø ØC ØØ ØA 64 6F 73 61 64 2E 61 73 6D 8C 96
[U75]     2: Ø7 ØØ ØØ Ø4 43 4F 44 45 44 98 Ø7 ØØ 6Ø DA Ø2 Ø2
[K95]     3: Ø1 Ø1 21 9Ø 77 ØØ ØØ Ø1 Ø6 47 45 54 5Ø 4F 49 E6
[X75]     4: Ø1 ØØ Ø6 57 52 49 54 45 48 C3 Ø1 ØØ Ø6 53 45 54
[Z95]     5: 5Ø 4F 49 Ø8 Ø2 ØØ Ø7 44 45 4C 45 54 45 48 2C Ø2
[FC5]     6: ØØ Ø7 4B 4F 5Ø 49 45 52 45 C3 Ø2 ØØ Ø4 4D 41 4B
[5D5]     7: 45 41 Ø1 ØØ ØA 52 45 41 44 53 45 4B 54 4F 52 6D
[785]     8: Ø2 ØØ Ø4 4F 5Ø 45 4E ØØ Ø1 ØØ Ø5 52 45 41 44 48
[EE5]     9: AØ Ø1 ØØ ØB 57 52 49 54 45 53 45 4B 54 4F 52 98
[H85]    1Ø: Ø2 ØØ Ø6 43 4C 4F 53 45 48 82 Ø1 ØØ 56 88 Ø4 ØØ
[M45]    11: ØØ A2 Ø1 D1 A2 ØE ØØ Ø1 ØØ ØØ ØØ Ø1 Ø1 ØØ Ø1 ØØ
[ED5]    12: ØØ ØØ Ø1 ØØ 4B AØ DE Ø1 Ø1 ØØ Ø1 55 8B EC 1E ØE
[1N5]    13: Ø7 BF ØØ ØØ C5 5E ØC 8A ØF ØA C9 74 ØB 32 ED 43
[7L5]    14: 8A Ø7 26 88 Ø5 47 E2 F7 26 C6 Ø5 ØØ 8B 46 ØA BA
[5H5]    15: ØØ ØØ ØE 1F B4 3D CD 21 C4 7E Ø6 73 Ø6 26 C6 Ø5
[AL5]    16: Ø1 EB Ø4 26 C6 Ø5 ØØ 1F 5D CA ØA ØØ 55 8B EC 1E
[NP5]    17: ØE Ø7 BF ØØ ØØ C5 5E ØC 8A ØF ØA C9 74 ØB 32 ED
[UG5]    18: 43 8A Ø7 26 88 Ø5 47 E2 F7 26 C6 Ø5 ØØ 8B 4E ØA
[YL5]    19: BA ØØ ØØ ØE 1F B4 3C CD 21 C4 7E Ø6 73 Ø6 26 C6
[ZL5]    2Ø: Ø5 Ø1 EB Ø4 26 C6 Ø5 ØØ 1F 5D CA ØA ØØ 55 8B EC
[GK5]    21: 8B 5E ØA B4 3E CD 21 C4 7E Ø6 73 Ø7 26 C6 Ø5 Ø1
[BP5]    22: EB Ø5 9Ø 26 C6 Ø5 ØØ 5D CA Ø6 ØØ 55 8B EC 8B 5E
[RM5]    23: 1Ø 8B 4E ØE 1E C5 56 ØA B4 3F CD 21 C5 5E Ø6 73
[HL5]    24: Ø5 C6 Ø7 Ø1 EB Ø3 C6 Ø7 ØØ 1F 5D CA ØC ØØ 55 8B
[HN5]    25: EC 8B 5E 1Ø 8B 4E ØE 1E C5 56 ØA B4 4Ø CD 21 C5
[ZL5]    26: 5E Ø6 73 Ø5 C6 Ø7 Ø1 EB Ø3 C6 Ø7 ØØ 1F 5D CA ØC
[5P5]    27: ØØ 55 8B EC 8B 5E ØA B8 Ø1 42 33 C9 8B D1 CD 21
[FF5]    28: C4 7E Ø6 73 Ø6 26 C6 Ø5 Ø1 EB Ø4 26 C6 Ø5 ØØ 5D
[VN5]    29: CA Ø6 ØØ 55 8B EC 8B 5E ØE B8 ØØ 42 8B 56 ØA 8B
[5H5]    3Ø: 4E ØC CD 21 C4 7E Ø6 73 Ø6 26 C6 Ø5 Ø1 EB Ø4 26
[ØL5]    31: C6 Ø5 ØØ 5D CA ØA ØØ 55 8B EC 1E ØE Ø7 BF ØØ ØØ
[3R5]    32: C5 5E ØA 8A ØF ØA C9 74 ØB 32 ED 43 8A Ø7 26 88
[ØK5]    33: Ø5 47 E2 F7 26 C6 Ø5 ØØ 8B 46 ØA BA ØØ ØØ ØE 1F
[EF5]    34: B4 41 CD 21 C4 7E Ø6 73 Ø6 26 C6 Ø5 Ø1 EB Ø4 26
[ØM5]    35: C6 Ø5 ØØ 1F 5D CA Ø8 ØØ 55 8B EC 8A 46 12 8B 56
[ØP5]    36: 1Ø 8B 4E ØE 1E C5 5E ØA 57 55 CD 25 58 5D 5F C5
[FM5]    37: 5E Ø6 72 Ø5 C6 Ø7 ØØ EB Ø3 C6 Ø7 Ø1 1F 8B E5 5D
[DM5]    38: CA ØE ØØ 55 8B EC 8A 46 12 8B 56 1Ø 8B 4E ØE 1E
[FN5]    39: C5 5E ØA 57 55 CD 26 58 5D 5F C5 5E Ø6 72 Ø5 C6
[BP5]    4Ø: Ø7 ØØ EB Ø3 C6 Ø7 Ø1 1F 8B E5 5D CA ØE ØØ 55 8B
[1U5]    41: EC FC 1E 8B 4E Ø6 C5 76 Ø8 C4 7E ØC F3 A4 1F 8B
[CH5]    42: E5 5D CA ØA ØØ 28 9C 1F ØØ C4 Ø7 Ø4 Ø1 Ø1 C4 25
[G75]    43: Ø4 Ø1 Ø1 C4 48 Ø4 Ø1 Ø1 C4 66 Ø4 Ø1 Ø1 C5 33 Ø4
[L74]    44: Ø1 Ø1 C5 51 Ø4 Ø1 Ø1 29 8A Ø2 ØØ ØØ 74
```

Listing 13. »dosadd.obj« ist der Hexdump aus »dosadd.asm«

```
[VE5]     1: 8Ø ØA ØØ Ø8 6D 61 75 73 2E 61 73 6D 49 96 ØC ØØ
[X85]     2: ØØ Ø4 44 41 54 41 Ø4 43 4F 44 45 21 98 Ø7 ØØ 48
[B45]     3: ØØ ØØ Ø2 Ø1 Ø1 15 98 Ø7 ØØ 28 1F ØØ Ø3 Ø1 Ø1 15
[5F5]     4: 8C ØE ØØ ØB 4D 41 55 53 48 41 4E 44 4C 45 52 ØØ
[D85]     5: 27 9Ø 13 ØØ ØØ Ø2 ØC 46 49 52 53 54 48 41 4E 44
[8C5]     6: 4C 45 52 ØØ ØØ ØØ C9 88 Ø4 ØØ ØØ A2 Ø1 D1 AØ 23
[P75]     7: ØØ Ø2 ØØ ØØ 5Ø 53 51 52 57 56 55 Ø6 1E 5Ø 53 51
[6K5]     8: 52 B8 ØØ ØØ 8E D8 E8 ØØ ØØ 1F Ø7 5D 5E 5F 5A 59
[3F5]     9: 5B 58 CB 12 9C ØB ØØ C8 ØE Ø4 Ø1 Ø1 84 13 Ø6 Ø2
[872]    1Ø: Ø1 DD 8A Ø2 ØØ ØØ 74
```

Listing 14. »maus.obj«: die Maustreiber-Grundlagen

```
[HH5]     1: 8Ø ØC ØØ ØA 6E 65 75 69 6E 74 2E 61 73 6D 68 96
[695]     2: ØC ØØ ØØ Ø4 44 41 54 41 Ø4 43 4F 44 45 21 98 Ø7
[855]     3: ØØ 48 ØØ ØØ Ø2 Ø1 Ø1 15 98 Ø7 ØØ 28 F1 ØØ Ø3 Ø1
[VD5]     4: Ø1 43 8C 34 ØØ Ø6 4F 4C 44 49 4E 54 ØØ Ø2 4F 4E
[GA5]     5: ØØ Ø3 41 4E 46 ØØ Ø3 54 49 43 ØØ Ø3 4B 4F 52 ØØ
[X85]     6: Ø4 54 49 43 31 ØØ Ø4 42 49 4C 44 ØØ Ø4 4B 4F 52
[975]     7: 31 ØØ Ø4 53 45 43 53 ØØ AØ 9Ø ØD ØØ ØØ Ø2 Ø6 4E
[F95]     8: 45 55 49 4E 54 ØØ ØØ ØØ 88 88 Ø4 ØØ ØØ A2 Ø1 D1
[6G5]     9: AØ F5 ØØ Ø2 ØØ ØØ 5Ø 53 Ø6 1E B8 ØØ ØØ 8E D8 FE
[1D5]    1Ø: ØE ØØ ØØ 75 79 C6 Ø6 ØØ ØØ 12 FE ØE ØØ ØØ 75 Ø9
[PE5]    11: C6 Ø6 ØØ ØØ Ø5 FE Ø6 ØØ ØØ 1E Ø7 8E 1E ØØ ØØ BB
[NH5]    12: 98 ØØ 8Ø 3F 3A 75 Ø5 C6 Ø7 2Ø EB Ø3 C6 Ø7 3A 26
[7M5]    13: FE ØE ØØ ØØ 75 46 26 C6 Ø6 ØØ ØØ 3C BB 9C ØØ FE
[ZK5]    14: Ø7 8Ø 3F 39 76 36 C6 Ø7 3Ø 8Ø EB Ø2 FE Ø7 8Ø 3F
[1L5]    15: 36 75 29 C6 Ø7 3Ø 8Ø EB Ø4 FE Ø7 8Ø 3F 34 75 ØF
[VL5]    16: 8Ø 7F FE 32 75 Ø9 C6 Ø7 3Ø C6 47 FE 3Ø EB ØD 8Ø
[KM5]    17: 3F 39 76 Ø8 C6 Ø7 3Ø 8Ø EB Ø2 FE Ø7 Ø6 1F 8Ø 3E
[H85]    18: ØØ ØØ ØØ 74 58 FE ØE ØØ ØØ 75 52 C6 Ø6 ØØ ØØ 12
[VG5]    19: FE ØE ØØ ØØ 75 Ø9 C6 Ø6 ØØ ØØ Ø5 FE Ø6 ØØ ØØ 8B
[DM5]    2Ø: 1E ØØ ØØ 1E 8E 1E ØØ ØØ FE Ø7 8Ø 3F 39 76 2D C6
[VF5]    21: Ø7 3Ø 83 EB Ø2 FE Ø7 8Ø 3F 36 75 2Ø C6 Ø7 3Ø 83
[PL5]    22: EB Ø4 FE Ø7 8Ø 3F 39 76 13 C6 Ø7 3Ø 83 EB Ø2 FE
[UR5]    23: Ø7 8Ø 3F 36 75 Ø6 C6 Ø7 3Ø FE 47 FC 1F 9C FF 1E
[DH5]    24: ØØ ØØ 1F Ø7 5B 58 CF AA 9C 5B ØØ C8 Ø5 Ø4 Ø1 Ø1
[WA5]    25: C4 ØB Ø6 Ø1 Ø4 C4 11 Ø6 Ø1 Ø4 C4 16 Ø6 Ø1 Ø5 C4
[ØC5]    26: 1C Ø6 Ø1 Ø5 C4 21 Ø6 Ø1 Ø4 C4 27 Ø6 Ø1 Ø7 C4 3C
[CA5]    27: Ø6 Ø1 Ø9 C4 43 Ø6 Ø1 Ø9 C4 8A Ø6 Ø1 Ø2 C4 91 Ø6
[2D5]    28: Ø1 Ø6 C4 97 Ø6 Ø1 Ø6 C4 9C Ø6 Ø1 Ø8 C4 A2 Ø6 Ø1
[XF5]    29: Ø8 C4 A7 Ø6 Ø1 Ø6 C4 AB Ø6 Ø1 Ø3 C4 BØ Ø6 Ø1 Ø7
[BK3]    3Ø: C4 EA Ø6 Ø1 Ø1 6C 8A Ø2 ØØ ØØ 74
```

Listing 15. »neuint.obj« für die Erzeugung von »sbackup.exe«

```
[HM5]     1: 8Ø ØD ØØ ØB 73 74 72 69 6E 67 73 2E 61 73 6D EF
[DA5]     2: 96 Ø7 ØØ ØØ Ø4 43 4F 44 45 44 98 Ø7 ØØ 6Ø AF Ø6
[B75]     3: Ø2 Ø1 Ø1 48 9Ø FE ØØ ØØ Ø1 Ø6 55 5Ø 43 48 41 52
[W95]     4: 37 Ø2 ØØ Ø6 49 4E 54 56 41 4C 59 Ø3 ØØ Ø6 5Ø 52
[8C5]     5: 45 46 49 58 A3 Ø2 ØØ Ø6 53 55 46 46 49 58 D6 Ø2
[285]     6: ØØ Ø6 41 53 54 59 5Ø 45 3Ø ØØ ØØ Ø6 41 4C 53 54
[TB5]     7: 59 5Ø 9Ø Ø6 ØØ Ø6 49 4E 54 53 54 52 D6 Ø4 ØØ Ø7
[8D5]     8: 44 45 4C 4C 45 46 54 82 Ø4 ØØ Ø7 57 4F 52 44 56
[VC5]     9: 41 4C F5 Ø3 ØØ Ø7 57 4F 52 44 53 54 52 1D Ø5 ØØ
[3C5]    1Ø: Ø8 53 43 41 4E 4C 45 46 54 51 Ø5 ØØ Ø8 44 45 4C
[DB5]    11: 52 49 47 48 54 B8 Ø4 ØØ Ø8 41 53 53 54 52 49 4E
[3C5]    12: 47 1A ØØ ØØ Ø8 4C 4F 53 54 52 49 4E 47 7A Ø2 ØØ
[X95]    13: Ø8 55 5Ø 53 54 52 49 4E 47 51 Ø2 ØØ Ø9 53 43 41
[LK5]    14: 4E 52 49 47 48 54 D7 Ø5 ØØ Ø9 56 4F 52 4B 4F 4D
[6D5]    15: 4D 45 4E 1C Ø6 ØØ Ø9 53 55 42 53 54 52 49 4E 47
[LE5]    16: ØF Ø3 ØØ ØA 47 45 54 5Ø 4F 53 46 45 4C 44 5A Ø6
[9A5]    17: ØØ Ø4 4C 45 46 54 3C ØØ ØØ Ø4 43 4F 5Ø 59 9Ø Ø5
[MD5]    18: ØØ Ø5 52 49 47 48 54 6C Ø1 ØØ Ø6 4C 4F 43 48 41
[P75]    19: 52 44 Ø2 ØØ CA 88 Ø4 ØØ ØØ A2 Ø1 D1 AØ 7Ø ØØ Ø1
[MR5]    2Ø: ØØ ØØ 55 8B EC 1E FC 32 ED C4 7E Ø4 C5 76 Ø8 AC
[3T5]    21: 8A C8 AA F3 A4 1F 8B E5 5D C2 Ø8 ØØ 55 8B EC C4
[4K5]    22: 7E Ø6 Ø6 57 C4 7E ØA Ø6 57 E8 ØØ ØØ 8B E5 5D CA
[VM5]    23: Ø4 ØØ 55 8B EC C4 46 Ø6 8C C2 5D CA Ø4 ØØ 55 8B
[MR5]    24: EC 83 C5 Ø6 8B 4E ØØ 1E C5 76 Ø2 C4 7E Ø6 8A 2C
[4S5]    25: 3A E9 73 Ø7 86 CD 32 ED 89 4E ØØ FC 32 ED 41 F3
[AN5]    26: A4 8B 4E ØØ C5 5E Ø6 88 ØF 1F 5D CA Ø6 ØØ 7Ø 9C
[9B5]    27: Ø6 ØØ 84 28 Ø4 Ø1 Ø1 AC A2 ØE ØØ Ø1 6C ØØ ØØ Ø1
[365]    28: Ø1 ØØ Ø1 ØØ ØØ ØØ Ø1 ØØ DF AØ ØØ Ø4 Ø1 6C Ø1 55
[CS5]    29: 8B EC 83 C5 Ø6 FD 8B 4E ØØ 32 ED 1E C5 76 Ø2 C4
[FG5]    3Ø: 7E Ø6 8A Ø4 32 E4 3B C1 73 Ø5 8B C8 89 4E ØØ 4Ø
[PR5]    31: 41 Ø3 FØ Ø3 F9 F3 A4 8B 4E ØØ C5 5E Ø6 88 ØF 1F
[7M5]    32: 5D CA Ø6 ØØ 3C 61 72 46 3C 7A 77 Ø4 2C 2Ø EB 3E
[6P5]    33: 3C 81 75 Ø4 BØ 9A EB 36 3C 84 75 Ø4 BØ 8E EB 2E
[AM5]    34: 3C 94 75 Ø4 BØ 99 EB 26 3C 82 75 Ø4 BØ 9Ø EB 1E
[9N5]    35: 3C 87 75 Ø4 BØ 8Ø EB 16 3C 86 75 Ø4 BØ 8F EB ØE
[EL5]    36: 3C A4 75 Ø4 BØ A5 EB Ø6 3C 91 75 Ø2 BØ 92 C3 3C
[AG5]    37: 41 72 46 3C 5A 77 Ø4 Ø4 2Ø EB 3E 3C 9A 75 Ø4 BØ
[CL5]    38: 81 EB 36 3C 8E 75 Ø4 BØ 84 EB 2E 3C 99 75 Ø4 BØ
[8M5]    39: 94 EB 26 3C A5 75 Ø4 BØ A4 EB 1E 3C 8F 75 Ø4 BØ
[KG5]    4Ø: 86 EB 16 3C 92 75 Ø4 BØ 91 EB ØE 3C 9Ø 75 Ø4 BØ
[RK5]    41: 82 EB Ø6 3C 8Ø 75 Ø2 BØ 87 C3 55 8B EC 8B 46 Ø6
[FG5]    42: E8 ØØ ØØ 5D CA Ø2 ØØ 55 8B EC 8B 46 Ø6 E8 ØØ ØØ
[PP5]    43: 5D CA Ø2 ØØ 55 8B EC 83 C5 Ø6 1E C5 5E ØØ C4 7E
[BL5]    44: Ø4 8A ØF 26 88 ØD 32 ED ØA C9 74 ØC 43 47 8A Ø7
[4P5]    45: E8 ØØ ØØ 26 88 Ø5 E2 F4 1F 5D CA Ø4 ØØ 55 8B EC
[HK5]    46: 83 C5 Ø6 1E C5 5E ØØ C4 7E Ø4 8A ØF 26 88 ØD 32
[CK5]    47: ED ØA C9 74 ØC 43 47 8A Ø7 E8 ØØ ØØ 26 88 Ø5 E2
[GP5]    48: F4 1F 5D CA Ø4 ØØ 55 8B EC 83 C5 Ø6 1E C5 5E ØØ
[EM5]    49: C4 7E Ø4 8A ØF 26 8A 2D ØA C9 74 11 3A E9 72 15
[6F5]    5Ø: 32 ED 43 47 8A Ø7 26 3A Ø5 75 ØA E2 F5 B8 Ø1 ØØ
[AU5]    51: 1F 5D CA Ø8 ØØ 33 CØ EB F7 55 8B EC 83 C5 Ø6 1E
[ØN5]    52: C5 5E ØØ C4 7E Ø4 8A ØF 26 8A 15 ØA C9 74 17 3A
[4L5]    53: D1 72 1B 32 F6 32 ED Ø3 FA Ø3 D9 8A Ø7 26 3A Ø5
[ZL5]    54: 75 ØC 4B 4F E2 F5 B8 Ø1 ØØ 1F 5D CA Ø8 ØØ 33 CØ
[1S5]    55: EB F7 55 8B EC 83 C5 Ø6 1E C5 5E ØØ C4 7E Ø4 8A
[HM5]    56: ØF 26 8A 15 32 ED 8A F5 8A E5 ØA C9 74 2A 43 47
[WG5]    57: 3A D1 72 24 FE CA 78 2Ø 57 53 51 8A Ø7 26 3A Ø5
[7N5]    58: 75 11 47 43 E2 F5 59 5B 5F 8B C7 2B 46 Ø4 1F 5D
[DU5]    59: CA Ø8 ØØ 59 5B 5F EB D7 33 CØ EB F2 55 8B EC 83
[LS5]    6Ø: C5 Ø6 C4 7E ØØ 1E C5 5E Ø4 8A ØF 32 ED 8A D5 43
[RH5]    61: 8A Ø7 3C 2D 75 Ø6 8Ø F2 Ø1 EB Ø5 9Ø 3C 2B 75 Ø7
[6P5]    62: E2 ED 33 CØ EB 5F 9Ø 3C 3Ø 72 65 3C 39 77 61 3C
[VP5]    63: 3Ø 75 ØA 43 8A Ø7 E2 EF 33 CØ EB 49 9Ø Ø3 D9 4B
[LE5]    64: 8Ø F9 Ø5 77 4B 52 BE Ø1 ØØ 33 CØ 5Ø 8A Ø7 3C 3Ø
[BL5]    65: 72 3C 3C 39 77 38 2C 3Ø 32 E4 F7 E6 ØB D2 75 2E
[EV5]    66: 5A Ø3 C2 72 2A 5Ø B8 ØA ØØ F7 E6 8B FØ 4B E2 DC
[ØG5]    67: 58 5A ØA D2 74 ØA 3D ØØ 8Ø 77 15 F7 D8 EB Ø6 9Ø
[5K5]    68: A9 ØØ 8Ø 75 ØB 26 C6 Ø5 ØØ 1F 5D CA Ø8 ØØ 58 58
[PN5]    69: 26 C6 Ø5 Ø1 33 CØ EB F1 55 8B EC 83 C5 Ø6 C4 7E
[ZP5]    7Ø: ØØ 1E C5 5E Ø4 8A ØF 32 ED 8A D5 43 8A Ø7 3C 2D
[6H5]    71: 75 Ø6 8Ø F2 Ø1 EB Ø5 9Ø 3C 2B 75 Ø7 E2 ED 33 CØ
[ZG5]    72: EB 5Ø 9Ø 3C 3Ø 72 56 3C 39 77 52 3C 3Ø 75 ØA 43
[6T5]    73: 8A Ø7 E2 EF 33 CØ EB 3A 9Ø Ø3 D9 4B 8Ø F9 Ø5 77
[GF5]    74: 3C 52 BE Ø1 ØØ 33 CØ 5Ø 8A Ø7 3C 3Ø 72 2D 3C 39
[KM5]    75: 77 29 2C 3Ø 32 E4 F7 E6 ØB D2 75 1F 5A Ø3 C2 72
[ZU5]    76: 1B 5Ø B8 ØA ØØ F7 E6 8B FØ 4B E2 DC 58 5A ØA D2
[PH5]    77: 75 ØB 26 C6 Ø5 ØØ 1F 5D CA Ø8 ØØ 58 58 26 C6 Ø5
[SP5]    78: Ø1 33 CØ EB F1 55 8B EC 83 C5 Ø6 8B 56 ØØ 1E C5
[ZN5]    79: 5E Ø2 8A ØF 3A CA 73 Ø2 8A D1 32 F6 43 8B FB Ø3
[6P5]    8Ø: FA 2A CA 8A D1 74 ØA 32 ED 8A Ø5 88 Ø7 47 43 E2
[AN5]    81: F8 8B 5E Ø2 88 17 1F 5D CA Ø6 ØØ 55 8B EC 83 C5
[WK5]    82: Ø6 8B 56 ØØ 1E C5 5E Ø2 8A ØF 3A CA 73 Ø2 8A D1
[HT5]    83: 2A CA 88 ØF 1F 5D CA Ø6 ØØ 55 8B EC 8B 46 Ø6 1E
[9M5]    84: ØE 1F BB ØØ ØØ 33 D2 8B CA BE ØA ØØ A9 ØØ 8Ø 74
[LM5]    85: Ø4 B5 Ø1 F7 D8 F7 F6 8Ø C2 3Ø 88 17 4B FE C1 32
[LS5]    86: D2 ØB CØ 75 FØ ØA ED 74 Ø6 C6 Ø7 2D 4B FE C1 88
[MK5]    87: ØF ØE 53 C4 7E Ø8 Ø6 57 E8 ØØ ØØ 1F 5D CA Ø2 ØØ
[1P5]    88: 55 8B EC 8B 46 Ø6 1E ØE 1F BB ØØ ØØ 33 D2 BE ØA
[XS5]    89: ØØ 8B CA F7 F6 8Ø C2 3Ø 88 17 4B FE C1 32 D2 ØB
[YM5]    9Ø: CØ 75 FØ 88 ØF ØE 53 C4 7E Ø8 Ø6 57 E8 ØØ ØØ 1F
[9S5]    91: 5D CA Ø2 ØØ 55 8B EC 8B 56 Ø6 1E C5 5E Ø8 C4 7E
[ZK5]    92: ØC 8A ØF 32 ED ØA C9 74 25 43 47 A3 9C 35 ØØ 84
[RC5]    93: D2 ØØ Ø1 Ø1 A1 Ø1 84 DF ØØ Ø1 Ø1 EC Ø1 85 Ø2 ØØ
[A95]    94: Ø1 Ø1 A1 Ø1 85 2B ØØ Ø1 Ø1 EC Ø1 C7 74 ØØ Ø1 Ø1
[3H5]    95: 73 ØØ 87 AA Ø4 Ø1 Ø1 C7 BB ØØ Ø1 Ø1 72 ØØ 87 DE
[ØC5]    96: Ø4 Ø1 Ø1 D1 AØ 4B Ø1 Ø1 68 Ø5 8A Ø7 3A C2 74 15
[DP5]    97: 26 88 Ø5 E2 F3 8B 5E Ø8 8A ØF 8B 7E ØC 26 88 ØD
[VV5]    98: 1F 5D CA Ø6 ØØ 8B CF 8B 7E ØC 2B CF 49 26 88 ØD
[5V5]    99: EB EE 55 8B EC 1E 8B 56 Ø6 8B 4E Ø8 C5 5E ØA BF
[7K5]   1ØØ: ØØ ØØ ØE Ø7 32 E4 8A F4 8A EC 3A CA 77 19 3A 17
[KL5]   1Ø1: 77 15 ØA C9 74 11 Ø3 CB Ø3 DA 8A Ø7 26 88 Ø5 FE
[YL5]   1Ø2: C4 4B 4F 3B D9 73 F3 26 88 25 ØE 57 C4 7E ØE Ø6
[KM5]   1Ø3: 57 E8 ØØ ØØ 1F 5D CA Ø8 ØØ 55 8B EC 8B 56 Ø6 1E
[AV5]   1Ø4: C5 5E Ø8 8A ØF 8A F1 32 ED ØA C9 74 2A Ø3 D9 ØE
[LF5]   1Ø5: Ø7 BF ØØ ØØ Ø3 F9 8A Ø7 3A C2 74 1B 26 88 Ø5 4B
[PR5]   1Ø6: 4F E2 F3 8A Ø7 26 88 Ø5 Ø6 57 C4 7E ØC Ø6 57 E8
[UM5]   1Ø7: ØØ ØØ 1F 5D CA Ø6 ØØ 2A F1 26 88 35 EB EA 55 8B
[SP5]   1Ø8: EC 8B 56 Ø6 1E C5 5E Ø8 BF ØØ ØØ ØE Ø7 32 F6 8A
[HV5]   1Ø9: ØF ØA C9 74 19 8A EE 8A E6 47 43 FE C4 8A Ø7 3A
[7N5]   11Ø: C2 75 Ø6 26 88 25 FE C6 47 E2 EF BF ØØ ØØ 26 88
[NN5]   111: 35 89 7E Ø8 8C 4E ØA 1F 5D CA Ø2 ØØ 55 8B EC C4
[WV5]   112: 7E Ø6 8B 56 ØA 1E C5 5E ØC 32 F6 8A ØF ØA C9 74
[MR5]   113: 19 8A EE 8A E6 47 43 FE C4 8A Ø7 3A C2 75 Ø6 26
[SU5]   114: 88 25 FE C6 47 E2 EF 8B 7E Ø6 26 88 35 1F 5D CA
[VV5]   115: ØA ØØ 55 8B EC 1E C4 7E Ø6 C5 5E ØA 8A ØF 32 ED
[UL5]   116: 26 88 ØD 43 47 8A Ø7 26 88 Ø5 E2 F7 1F 5D CA Ø8
[H85]   117: ØØ BF 9C 27 ØØ C4 36 ØØ Ø1 Ø1 6B Ø1 84 68 Ø4 Ø1
[L85]   118: Ø1 C4 88 ØØ Ø1 Ø1 6C ØØ 84 A6 Ø4 Ø1 Ø1 C4 BF ØØ
[795]   119: Ø1 Ø1 6C ØØ C4 E2 ØØ Ø1 Ø1 6C ØØ F4 8A Ø2 ØØ ØØ
[A8Ø]   12Ø: 74
```

Listing 16. »strings.obj« wird in »sbackup.exe« eingebaut

```
[1N5]     1: 8Ø ØD ØØ ØB 77 69 6E 64 6F 77 73 2E 61 73 6D EE
[D75]     2: 96 Ø7 ØØ ØØ Ø4 43 4F 44 45 44 98 Ø7 ØØ 6Ø 36 ØØ
[A85]     3: Ø2 Ø1 Ø1 C7 9Ø 23 ØØ ØØ Ø1 Ø6 4F 46 46 53 45 54
[1A5]     4: ØØ ØØ ØØ Ø6 43 55 52 53 4F 52 24 ØØ ØØ Ø8 4E 4F
[3A5]     5: 43 55 52 53 4F 52 2D ØØ ØØ C7 88 Ø4 ØØ ØØ A2 Ø1
[7G5]     6: D1 AØ 3A ØØ Ø1 ØØ ØØ 55 8B EC 8B 46 Ø6 32 E4 8B
[RM5]     7: 56 Ø8 32 F6 48 4A B1 Ø7 8B D8 D3 EØ 93 B1 Ø5 D3
[SN5]     8: EØ Ø3 C3 D1 E2 Ø3 C2 5D CA Ø4 ØØ B9 Ø9 Ø8 B8 ØØ
[TK5]     9: Ø1 CD 1Ø CB B9 3Ø 3Ø B8 ØØ Ø1 CD 1Ø CB F1 8A Ø2
[GVØ]    1Ø: ØØ ØØ 74
```

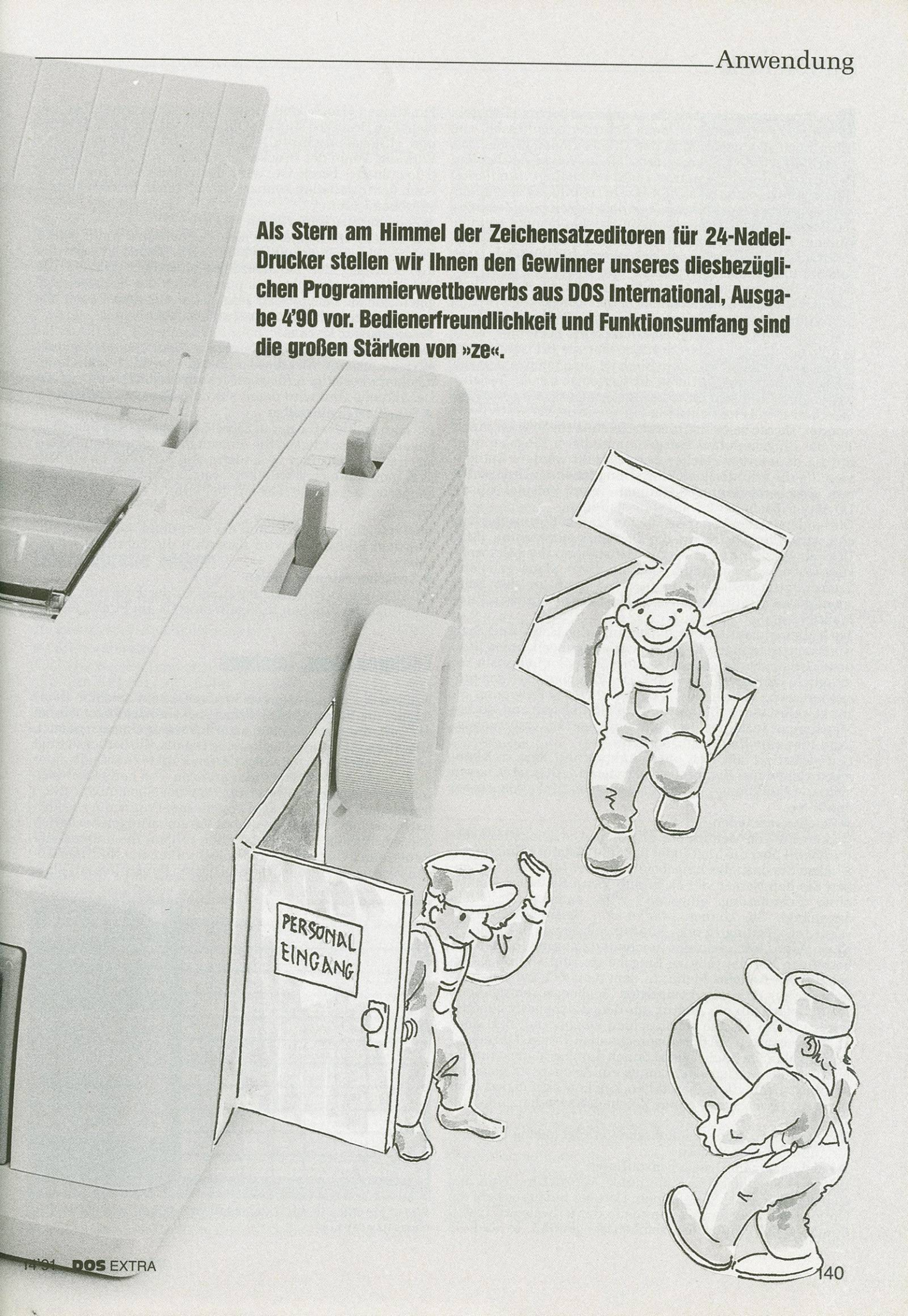
Listing 17. »windows.obj«: die Fenster- und Menüfunktionen

Turbo C 2.0

Tanz der Nadeln



Als Stern am Himmel der Zeichensatzeditoren für 24-Nadel-Drucker stellen wir Ihnen den Gewinner unseres diesbezüglichen Programmierwettbewerbs aus DOS International, Ausgabe 4'90 vor. Bedienerfreundlichkeit und Funktionsumfang sind die großen Stärken von »ze«.



Das Programm »ze« ist ein Zeichensatzeditor für 24-Nadel-Drucker, mit dem sich neue Schriften komfortabel am Bildschirm erzeugen und in den Drucker laden lassen. Das Programm wurde so konzipiert, daß ein Anwender bei der Bedienung des Programms nicht unbedingt fundiertes Fachwissen benötigt. Listing 1 (»ze.c«) beinhaltet zusätzlich eine Konvertierungsfunktion, mit dem bestehende Zeichensätze aus diversen Public-Domain-Dateien benutzt werden können. Der Zeichensatzeditor ist für einen 24-Nadel-Matrixdrucker konzipiert, der den Epson-ESC/P-Befehlssatz versteht und Zeichensätze laden und aktivieren kann. Sofern Ihr Drucker diese Fähigkeit momentan noch nicht hat, gibt es vielleicht vom Hersteller Erweiterungskarten (RAM-Einschübe), die dem Drucker eine neue Welt erschließen. Weiterhin wird eine Hercules-Grafikkarte, EGA- oder VGA-Grafikkarte (mindestens 128 KByte) für die Bildschirmdarstellung vorausgesetzt. Schließlich ist auch noch eine Maus erforderlich, da das Editieren der Fontdaten mit der Tastatur wenig sinnvoll erscheint. Als Druckerschnittstelle wird die Standard-Druckerschnittstelle (»prn«, meistens »lpt1«) verwendet. Da die selbstdefinierten Zeichen im Drucker einen Teil des Eingabepuffers belegen, muß dieser Eingabepuffer für die definierten Zeichen bereitgestellt werden. Entnehmen Sie die erforderliche Einstellung für Ihren Drucker Ihrem Druckerbedienungshandbuch (zum Beispiel Epson LQ 500: Dipschalter 2.5 auf »off« stellen).

Die beiden letzten Voraussetzungen für den Einsatz des Zeichensatzeditors betreffen die Betriebssystemversion (MS-DOS ab Version 3.0) und den Maustreiber, der Microsoft-kompatibel arbeiten muß. Der mit Ihrer Maus gelieferte Treiber, zum Beispiel »mouse.sys«, muß in der Datei »config.sys« installiert sein (siehe Installationshinweis Ihres Maustreibers).

Nach dieser detaillierten Aufstellung der Hard- und Softwarevoraussetzungen kommen wir zum angenehmen Teil, dem Einsatz des Zeichensatzeditors. Zum Starten geben Sie lediglich »ze« und [Enter] ein. Der Zeichensatzeditor präsentiert sich, wie in Bild 1 dargestellt. Das Programm erkennt selbständig die verwendete Grafikkarte und ob ein Maustreiber installiert ist. Fehlt eine der Voraussetzungen, führt dies zum Programmabbruch.

Grundsätzlich läßt sich eine Funktion mit dem in Klammern stehenden Buchstaben aktivieren. Alternativ lassen sich alle Funktionen auch mit der Maus (linke Maustaste) anwählen.

● Zeichensatz laden

Wählen Sie den Menüpunkt »(L)aden«. Die Zeichensatzauswahl wird über die Maus oder über die Tastatur gehandhabt. Klicken Sie dazu das Eingabefeld mit der Maus an, und geben Sie den Namen der Datei ein. Wenn Sie einen Namen ohne Erweiterung eingeben, wird automatisch ».fnt« angehängt. Sie können hierbei den Beispielzeichensatz »caps.fnt« (Listing 6) oder künftig selbsterzeugte Zeichensätze laden. Nachdem der Zeichensatz ausgewählt wurde, bestätigen Sie die richtige Eingabe im Feld »(O)k«. Es erscheint ein weiteres Menü, in dem man sich zwischen einem Auszug oder dem kompletten Zeichensatz entscheiden kann. Sofern nur ein Auszug aus dem Zeichensatz benötigt wird, klicken Sie die gewünschten Zeichen mit der Maus an, oder drücken Sie die entsprechenden Tasten. Versehentlich angeklickte Buchstaben können durch erneutes Anwählen gelöscht werden. War bereits ein Zeichensatz geladen, so werden nur die angewählten Zeichen aktualisiert. Auf diese Art lassen sich mehrere Zeichensätze miteinander mischen.

Das Speichern eines Zeichensatzes erfolgt analog über den Menüpunkt »(S)peichern«.

● Zeichensatz im Drucker installieren

Wählen Sie zuerst den Menüpunkt »(D)rucker«. Nun läßt sich der Druckeffekt bestimmen. Darüber hinaus besteht die Option, den Zeichensatz komplett oder teilweise zu installieren. Ist der Proportionalzeichensatz gewählt, so sind die Funktionen »Hoch- und Tiefstellung« nicht anwählbar. Bei manchen Druckern sind die Optionen »(U)mrahmte Schrift« und »(S)chattenschrift« (»Outline« und »Shadow«) ohne Wirkung. Wenn der Drucker zum Laden von Zeichensätzen (»download«) bereit ist, wird der Zeichensatz installiert. Nach der Installation können Sie sich einen Probeausdruck ausgeben lassen.

● Zeichensätze importieren

Mit dieser Funktion lassen sich Zeichensätze von diversen Public-Domain-Softfont-Dateien lesen. Zuerst müssen Sie die Schriftart des Zeichensatzes einstellen. Wählen Sie dann den Menüpunkt »(I)mport«. Nach der Selektion des gewünschten Dateiformats fahren Sie mit dem Laden des Zeichensatzes wie zuvor beschrieben fort.

● Zeichensatz löschen

Zum Löschen eines Zeichensatzes dient der Menüpunkt »(N)eu«. Sofern Sie diese Funktion wirklich aktivieren möchten, bestätigen Sie die Sicherheitsabfrage mit »(J)a«. Der aktuelle Font wird damit aus dem Speicher entfernt.

● Schriftarten einstellen

Sie können hier eine der in Tabelle 1 dargestellten Schriftarten auswählen. Klicken Sie einfach das entsprechende Feld an, um die Schriftart zu ändern. Auf MS-DOS-Ebene läßt sich ganz einfach feststellen, welche Schriftart eine Zeichensatzdatei enthält. Geben Sie hierfür:

```
type caps.fnt
```

im Verzeichnis mit »ze« und der oder den Fontdateien ein. Auf dem Bildschirm wird daraufhin die Schriftart ausgegeben.

● Zwischenraum einstellen

Im Menüpunkt »(Z)wischenraum« ist der Freiraum rechts und links der Zeichen in Pixeln (jeweils ein Punkt) festzulegen.

Editieren eines Zeichens

Wählen Sie mit der Maus im linken Feld ein Zeichen. Es erscheint der Editor (Bild 2). Prinzipiell sind der Phantasie bei der Gestaltung von Zeichen natürlich wenig Grenzen gesetzt, allerdings gibt es zwei technische Details, die berücksichtigt werden müssen. Zum einen dürfen Punkte innerhalb einer Zeile nicht direkt aneinander grenzen, das heißt, daß zwischen zwei horizontal gesetzten Punkten eine Punktposition frei bleiben muß. Dies wird von diesem Programm automatisch beachtet. Weiterhin gibt es Vereinbarungen bezüglich der Schriftgröße, die Sie einhalten sollten: Die unteren vier Punktreihen (bis zur »Grundlinie«) sollten nur für Unterlängen verwendet werden. Die drei oberen Punkte (bis zur Linie

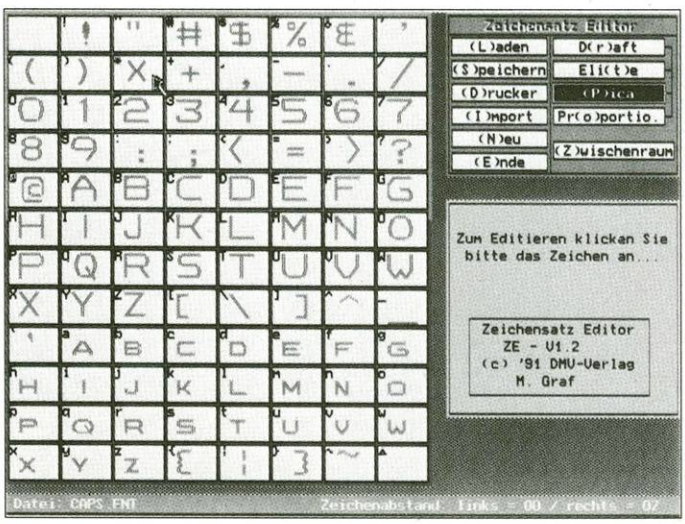


Bild 1. Ein Blick in das Hauptmenü von »ze« mit geladenem Zeichensatz »caps.fnt«

»Großbuchstaben«) werden von »normalen« Großbuchstaben nicht benötigt. Bei Zeichen, die zur Hoch- oder Tiefstellung verwendet werden, müssen Sie sich an die gegebenen Begrenzungslinien halten.

● Zeichenfarbe ändern

Mit der Option »(S)chwarz« oder »(W)eiß« wählen Sie die Zeichenfarbe (an der Farbe des Mauscursors läßt sich dies erkennen). Bei gedrückter linker Maustaste können die Pixel (Einzelpunkte) des Zeichens im Editierfeld in der gewünschten »Farbe« gesetzt werden.

● Ablage

Mit der Funktion »a(b)legen« wird das Zeichen in einen Zwischenspeicher gelegt, der bis zum nächsten Ablegen erhalten bleibt. Entsprechend dient die Funktion »(h)olen« zum Laden des Zeichens, das sich gerade im Zwischenspeicher befindet. »(t)auschen« ersetzt das Zeichen im Editierfeld mit dem Zeichen in der Ablage und umgekehrt.

● Zeichen verschieben

Klicken Sie den Pfeil an, und das Zeichen wird in die gewählte Richtung um einen Pixel verschoben.

● Zeichen drehen

Das aktuelle Zeichen wird mit der Funktion »(d)rehen« um 90 Grad gegen den Uhrzeigersinn gedreht.

● Zeichen löschen

Die Funktion »(N)eu« löscht das Editierfeld.

Zeichenbreite ändern

Die Zeichenbreite ist nur bei Proportionalschrift einstellbar. Klicken Sie hierzu im Balken neben »Zeichenbreite:« die gewünschte Zeichenbreite an.

● Editor verlassen

Nach der Bearbeitung eines Zeichens kann durch den Menüpunkt »(F)ertig« der Editor verlassen werden. Das editierte Zeichen wird damit in den Zeichensatz eingefügt (Bild 3). »(A)bbruch« verläßt den Editor, ohne daß der Zeichensatz verändert wird.

Schriftart	Auflösung
Draft	9x24 Punkte
Elite	23x24 Punkte
Pica	29x24 Punkte
Proportional	bis 37x24 Punkte

Tabelle 1. Die Übersicht enthält eine Aufstellung der mit »ze« wählbaren Schriftarten und zugehörigen Auflösungen

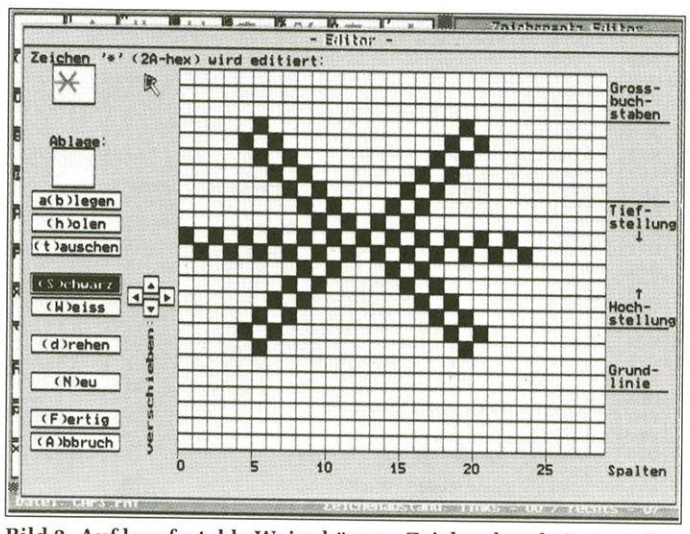


Bild 2. Auf komfortable Weise können Zeichen bearbeitet werden

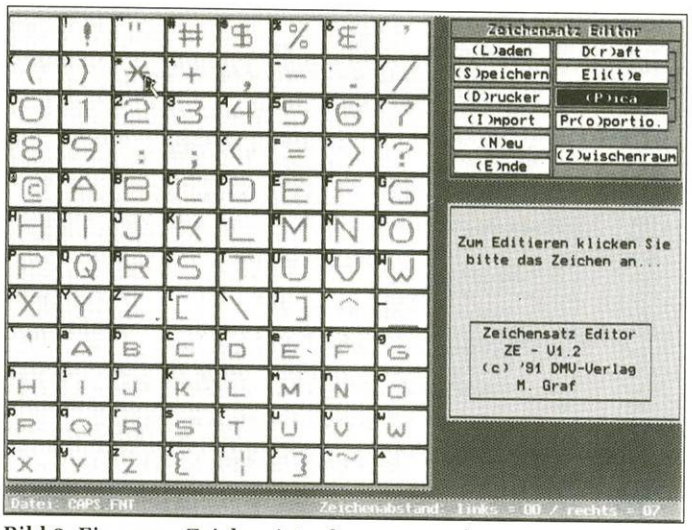


Bild 3. Ein neues Zeichen ist geboren

● Programm beenden

Sie wählen den Menüpunkt »(E)nde« und bestätigen das Programmende mit »(J)a«.

● Dateiformate

»ze« setzt als Endung für die Fontdateien »fnt« voraus. Die Datei enthält Informationen über die ASCII-Zeichen 32 bis 127. Die ersten 12 Byte enthalten dabei den ASCII-Text, der die Schriftart angibt, also »Draft«, »Elite«, »Pica« oder »Proportional«. Besteht der Name der Schriftart aus weniger als 12 Zeichen, wird der Rest mit Leerzeichen aufgefüllt. Das nächste Byte (Position 13) enthält den Dezimalcode 0. Darauf folgt ein Byte mit dem Dezimalcode 26 ([Ctrl-Z] dient dem Abbruch der Textausgabe beim Type-Befehl). Das 15. Byte enthält den Freiraum in Punktspalten links des einzelnen Zeichens. Die nächsten 96 Byte geben die Breite des entsprechenden Zeichens in Pixel an, das heißt, das erste Byte gibt die Breite des ersten ASCII-Zeichens an etc. Als nächstes folgt ein Byte, das den Freiraum in Punktspalten rechts der einzelnen Zeichen angibt. Die nun folgenden Daten enthalten die eigentlichen Informationen über das Aussehen der Zeichen, die in der Reihenfolge des ASCII-

Schriftart	Anzahl der Byte je Zeichen
Draft	27 Byte
Elite	69 Byte
Pica	87 Byte
Proportional	111 Byte

Tabelle 2. Die verschiedenen Schriftarten benötigen unterschiedlich viele Byte in der Datei zur Definition jedes einzelnen Zeichens

Codes gespeichert sind. Welche Schriftart wieviele Byte benötigt, geht aus Tabelle 2 hervor. Jede Punktspalte eines Zeichens besteht aus 24 Punkten, die durch drei Byte repräsentiert werden. Byte 1 steht für die oberen acht Punkte, Byte 2 für die mittleren acht und Byte 3 für die unteren acht Punkte einer Spalte. Der Wert eines Bytes ergibt sich aus der Summe der Wertigkeiten der gesetzten Punkte.

Bei der Importfunktion ergeben sich Abweichungen zum zuvor beschriebenen Dateiformat. Die Namenserweiterung der Datei ist hier beliebig. In der importierten Datei sind zwar Informationen über die ASCII-Zeichen 33 bis 127 enthalten, allerdings keinerlei Auskunft über Schriftart, Zeichenbreite und Freiraum zwischen den Zeichen. Bezüglich des Aussehens der einzelnen Zeichen sind die Daten identisch mit dem Dateiformat von »ze«.

(Markus Graf/bj)

```
[YW3]    1: /* Zeichensatz Editor ZE
[4H8]    2: ____________________________________
[ØØØ]    3:
[D54]    4: (c) 1991 DOS International
[ØØØ]    5:
[LM4]    6: Autor.............: Markus Graf
[N93]    7: Version...........: 1.2
[VL4]    8: Zuerst erstellt...: 18.Ø3.199Ø
[SZ4]    9: Zuletzt geändert..: 19.Ø4.199Ø
[4H8]   1Ø: ____________________________________
[ØØØ]   11:
[ØØØ]   12:
[7K1]   13: Compiler:
[Y91]   14: =========
[4L3]   15: - Turbo C Compiler V2.Ø
[AN3]   16: - Speichermodel: Large
[U64]   17: - Environment: Tab Size = 2
[X35]   18: - Projectdatei : ZE.PRJ -> [ ZE.C
[W55]   19:   EGAVGA.OBJ   HERC.OBJ   LITT.OBJ ]
[ØØØ]   2Ø:
[VH5]   21: Erforderliche Hardware/Software:
[YV5]   22:     - Hercules- oder EGA-Karte >= 128Kbyte
[BY3]   23:     - Maus incl. Maustreiber
[ØØØ]   24:
[17Ø]   25: */
[ØØØ]   26:
[8W2]   27: #include<stdio.h>
[BE3]   28: #include<graphics.h>
[EL2]   29: #include<dir.h>
[B53]   3Ø: #include<string.h>
[7M2]   31: #include<dos.h>
[ML2]   32: #include<mem.h>
[2T2]   33: #include<alloc.h>
[RN2]   34: #include<math.h>
[XU2]   35: #include<ctype.h>
[7P2]   36: #include<bios.h>
[ØØØ]   37:
[ØØØ]   38:
[NK4]   39: /* Konstantendefinitionen */
[ØØØ]   4Ø:
[4N4]   41: #define VERSION    " ZE - V1.2 "
[ØØØ]   42:
[RH2]   43: #define ERROR   Ø
[8B2]   44: #define OK      -1
[6W2]   45: #define  NIX      1ØØØ
[ØØØ]   46:
[GD2]   47: #define DRAFT 1
[MD2]   48: #define ELITE 2
[GA2]   49: #define PICA  3
[ZS3]   5Ø: #define PROPORTIONAL 4
[XV3]   51: #define  ZWISCHENRAUM  5
[ØØØ]   52:
[KR2]   53: #define  LADEN     6
[B53]   54: #define SPEICHERN 7
[P23]   55: #define DRUCKER    8
[8Z2]   56: #define IMPORT    9
[CR2]   57: #define NEU        1Ø
[UN2]   58: #define ENDE      11
[ØØØ]   59:
[R53]   6Ø: #define SCHWARZ    12
[DV2]   61: #define WEISS      13
[723]   62: #define ABLEGEN    14
[HØ3]   63: #define  HOLEN      15
[V23]   64: #define  SPALTE    16
[943]   65: #define TAUSCHEN  17
[ØY2]   66: #define RECHTS    18
[VV2]   67: #define LINKS      19
[YØ3]   68: #define  DREHEN    2Ø
[ØØØ]   69:
[2G3]   7Ø: #define OUTLINE        16
[YC3]   71: #define  SHADOW        32
[RN3]   72: #define BREITDRUCK    64
[BY3]   73: #define  HOCHDRUCK      128
[9Y3]   74: #define  FETTDRUCK      256
[9L3]   75: #define  KURSIV        512
[MA4]   76: #define  HOCHSTELLUNG  1Ø24
[NB4]   77: #define  TIEFSTELLUNG  2Ø48
[A94]   78: #define  INSTALLIEREN  4Ø96
[7W3]   79: #define  KOMPLETT      8192
[M34]   8Ø: #define  TEILWEISE      16384
[ØØØ]   81:
[9W2]   82: #define  L_MIN    1
[CP2]   83: #define R_MIN    2
[8Z2]   84: #define  L_PLUS   3
[XZ2]   85: #define  R_PLUS   4
[ØØØ]   86:
[D55]   87: /* globale Variablendefinition */
[ØØØ]   88:
[EU2]   89: int schriftart;
[343]   9Ø: /* DRAFT, ELITE, ... */
[FY3]   91: unsigned drucker_mode;
[R14]   92: /* FETTDRUCK, KURSIV, ... */
[ØØØ]   93:
[124]   94: unsigned   markiert[96];
[ZU5]   95: /* alle markierten Zeichen = Øxffff */
[ØØØ]   96:
[8M2]   97: typedef struct
[P7Ø]   98:   {
[6X4]   99:     unsigned char posxy[37][3];
[ØZ5]  1ØØ: /* 3 Bytes vertikal, max. 37 Spalten */
[WW1]  1Ø1:   } z_bytes;
[MN3]  1Ø2: z_bytes zeichen[96];
[965]  1Ø3: /* Speicher für Zeichensatz */
[ØØØ]  1Ø4:
[8M2]  1Ø5: typedef struct
[P7Ø]  1Ø6:   {
[XS2]  1Ø7:     unsigned char dØ;
[MD3]  1Ø8:     unsigned char d1[96];
[ZS2]  1Ø9:     unsigned char d2;
[S91]  11Ø:   } form;
[P42]  111: form format;
[UØ7]  112: /* Breite des PROPORTIONAL Zeichensatzes */
[ØØØ]  113:
[RS2]  114: z_bytes ablage;
[5T6]  115: /* Variable enthält Zeichen der Ablage */
[DM3]  116: char    ablage_form = 1;
[BØ6]  117: /* enthält Zeichenbreite der Ablage */
[ØØØ]  118:
[HK7]  119: char datei[MAXPATH];            /* aktuelle
            Datei */
[TYB]  12Ø: char start_pfad[MAXPATH];       /* aktuell
            es Verzeichnis nach dem Start */
[ØØØ]  121:
[ØØØ]  122:
[529]  123: /***************************** Macro-Defin
            itionen ***********************/
[ØØØ]  124:
[C3C]  125: #define CLS_ZEICHEN(satz)    memset(satz,Ø,
            1Ø656);    /* Zeichensatz leeren */
[ØØØ]  126:
[HF9]  127: /***************************** Funktions-P
            rototypen *********************/
[ØØØ]  128:
[KL2]  129: void ende(void);
[3V2]  13Ø: void event(void);
[UK2]  131: void neu(void);
[874]  132: void zeichen_laden(void);
[GG6]  133: int  lese_datei(char *,z_bytes *,form *);
[G53]  134: void import(void);
[ZX5]  135: int   lese_import(char *,z_bytes *);
[MØ5]  136: void zeichen_speichern(void);
[KU4]  137: void speichere_datei(char *);
[X93]  138: void drucker(void);
[4N5]  139: void zeichen_satz_install(void);
[HH4]  14Ø: void zeichen_install(int);
[9E6]  141: void invert_drucker_taste(unsigned);
[376]  142: unsigned get_drucker_taste(int,int);
[SH5]  143: int  auswahl(char *,char *,int,int);
[954]  144: void zwischenraum(void);
[AG2]  145: void edit(int);
[1W6]  146: void set_raster_char(int,int,int,char *);
[AT6]  147: void set_edit_punkt(int,int,int,int);
[FW4]  148: void invert_edit_taste(int);
[LZ5]  149: unsigned get_edit_taste(int,int);
[496]  15Ø: void print_edit_char(int,int,char *);
[6H4]  151: int  initialisieren(void);
[L74]  152: void reset_format(void);
[S54]  153: void statuszeile(void);
[6W3]  154: void titelzeile(void);
[2A5]  155: int  get_menue_taste(int,int);
[ER3]  156: void invertwahl(int);
[V46]  157: void rahmen(int,int,int,int,char*);
[E57]  158: void zeichensatz_tabelle(z_bytes *,form);
[PL6]  159: void print_zeichen(int,int,char *,int);
[ØØØ]  16Ø:
[5K1]  161: /* UNITS */
[ØØØ]  162:
[KX4]  163: #include "G:\TC_PLUS\MOUSE.C"
[SB5]  164: #include "G:\TC_PLUS\GRAPHIK.C"
[9B5]  165: #include "G:\TC_PLUS\DRUCKER.C"
[655]  166: /* hier eigene Werte einsetzen */
[922]  167: /* Programm */
[ØØØ]  168:
[UWØ]  169: main()
[P7Ø]  17Ø: {
[4A3]  171:   if (initialisieren())
[P7Ø]  172:   {
[ZF2]  173:     statuszeile();
[162]  174:     titelzeile();
[1D6]  175:     zeichensatz_tabelle(zeichen,format);
[NBØ]  176:     do
[P7Ø]  177:     {
[D71]  178:       event();
[Ø41]  179:       ende();
[TV1]  18Ø:     } while(1);
[U7Ø]  181:   }
[U7Ø]  182: }
[ØØØ]  183:
[4U6]  184: /*.....................................
            ..............*/
[ØØØ]  185:
[RH2]  186: void ende(void)
[ØØØ]  187:
[DD3]  188: /* Programm beenden */
[P7Ø]  189: {
[AY2]  19Ø:   invertwahl(ENDE);
[V7A]  191:   if (box(12Ø,1ØØ,52Ø,2ØØ,"ENDE","Wollen Si
            e das Programm verlassen ?",
```

```
[PY5]  192:        "  (J)a  "," (N)ein ","",BLACK,LIGHTR
       ED)==1)
[P7Ø]  193:    {
[U12]  194:      mouse_off();
[S42]  195:      closegraph();
[8R4]  196:      setdisk(start_pfad[Ø]-'A');
[R23]  197:      chdir(start_pfad);
[C61]  198:      exit(1);
[U7Ø]  199:    }
[AY2]  2ØØ:    invertwahl(ENDE);
[U7Ø]  2Ø1: }
[ØØØ]  2Ø2:
[4U6]  2Ø3: /*.....................................
       ..............*/
[ØØØ]  2Ø4:
[DR2]  2Ø5: void event(void)
[ØØØ]  2Ø6:
[5SA]  2Ø7: /* wartet auf eine Menüwahl und startet die
        gewünschte Aktion */
[P7Ø]  2Ø8: {
[F15]  2Ø9:   int x,y,           /* Mausposition */
[RV3]  21Ø:       taste,         /* Maustaste */
[TL3]  211:       key,           /* Tastatur */
[7W5]  212:       gewaehlt;       /* gewählte Funktion *
       /
[XV6]  213:   void *picptr;       /* Bildzwischenspeiche
       r */
[ØØØ]  214:
[PØ2]  215:   mouse_on();
[NBØ]  216:   do
[P7Ø]  217:   {
[XZ1]  218:     gewaehlt = Ø;
[NBØ]  219:     do
[P7Ø]  22Ø:     {
[V24]  221:       mouse_info(&x,&y,&taste);
[FY2]  222:     } while(taste!=Ø);
[NBØ]  223:     do
[P7Ø]  224:     {
[V24]  225:       mouse_info(&x,&y,&taste);
[YN1]  226:       if (kbhit())
[P7Ø]  227:       {
[BR4]  228:         key = toupper(get_taste());
[5X1]  229:         switch(key)
[P7Ø]  23Ø:         {
[584]  231:           case 'L': gewaehlt = LADEN;
[PØ1]  232:                     break;
[D25]  233:           case 'S': gewaehlt = SPEICHERN;
[PØ1]  234:                     break;
[5G4]  235:           case 'D': gewaehlt = DRUCKER;
[PØ1]  236:                     break;
[BF4]  237:           case 'I': gewaehlt = IMPORT;
[PØ1]  238:                     break;
[AU3]  239:           case 'N': gewaehlt = NEU;
[PØ1]  24Ø:                     break;
[3Z3]  241:           case 'E': gewaehlt = ENDE;
[PØ1]  242:                     break;
[UA4]  243:           case 'R': gewaehlt = DRAFT;
[PØ1]  244:                     break;
[4A4]  245:           case 'T': gewaehlt = ELITE;
[PØ1]  246:                     break;
[LØ4]  247:           case 'P': gewaehlt = PICA;
[PØ1]  248:                     break;
[PL5]  249:           case 'O': gewaehlt = PROPORTIONAL
       ;
[PØ1]  25Ø:                     break;
[EL5]  251:           case 'Z': gewaehlt = ZWISCHENRAUM
       ;
[PØ1]  252:                     break;
[U7Ø]  253:         }
[U7Ø]  254:       }
[NH5]  255:     } while((taste==Ø) && (gewaehlt==Ø));
[AT7]  256:     if (!gewaehlt) gewaehlt = get_menue_tas
       te(x,y);
[DT2]  257:     switch(gewaehlt)
[P7Ø]  258:     {
[5K4]  259:       case LADEN: zeichen_laden();
[PØ1]  26Ø:                 break;
[EP2]  261:       case SPEICHERN:
[AC3]  262:                 zeichen_speichern();
[PØ1]  263:                 break;
[5V3]  264:       case DRUCKER: drucker();
[PØ1]  265:                 break;
[8W3]  266:       case IMPORT:  import();
[PØ1]  267:                 break;
[BT2]  268:       case NEU:    neu();
[PØ1]  269:                 break;
[ZY1]  27Ø:       case DRAFT:
[1Y1]  271:       case ELITE:
[1N1]  272:       case PICA:
[883]  273:       case PROPORTIONAL:
[T44]  274:                 invertwahl(schriftart);
[N34]  275:                 if (schriftart!=gewaehlt)
[P7Ø]  276:                 {
[9P3]  277:                   schriftart = gewaehlt;
[AP2]  278:                   reset_format();
[U12]  279:                   mouse_off();
[MZ6]  28Ø:                   picptr = getimage_in_pu
       ffer(413,12Ø,639,18Ø);
```

```
[LA8]  281:                   putbox(413,12Ø,639,18Ø,
       "","Bitte warten...",BLACK,GREEN);
[1D6]  282:                   zeichensatz_tabelle(zei
       chen,format);
[ØR5]  283:                   putimage(413,12Ø,picptr
       ,COPY_PUT);
[ØR2]  284:                   farfree(picptr);
[PØ2]  285:                   mouse_on();
[U7Ø]  286:                 }
[CR3]  287:                 invertwahl(gewaehlt);
[PØ1]  288:                 break;
[M63]  289:       case ZWISCHENRAUM:
[FN2]  29Ø:                 zwischenraum();
[PØ1]  291:                 break;
[U7Ø]  292:     }
[G48]  293:     if ((gewaehlt>=' ') && (gewaehlt<=127))
       edit(gewaehlt);
[ZF2]  294:     statuszeile();
[L84]  295:   } while(gewaehlt != ENDE);
[U7Ø]  296: }
[ØØØ]  297:
[4U6]  298: /*.....................................
       ..............*/
[ØØØ]  299:
[EB2]  3ØØ: void neu(void)
[ØØØ]  3Ø1:
[DZ3]  3Ø2: /* Zeichensatz löschen */
[P7Ø]  3Ø3: {
[ØT2]  3Ø4:   invertwahl(NEU);
[P47]  3Ø5:   if (box(413,15Ø,639,25Ø,"Zeichensatz loes
       chen",
[GY9]  3Ø6:         "Soll der Zeichensatz|geloescht wer
       den ?","  (J)a  "," (N)ein ","",
[S43]  3Ø7:         BLACK,LIGHTRED) == 1)
[P7Ø]  3Ø8:   {
[CB5]  3Ø9:     CLS_ZEICHEN(zeichen[Ø].posxy);
[AP2]  31Ø:     reset_format();
[1D6]  311:     zeichensatz_tabelle(zeichen,format);
[XX1]  312:     datei[Ø] = Ø;
[U7Ø]  313:   }
[ØT2]  314:   invertwahl(NEU);
[U7Ø]  315: }
[ØØØ]  316:
[4U6]  317: /*.....................................
       ..............*/
[ØØØ]  318:
[K34]  319: void zeichen_laden(void)
[ØØØ]  32Ø:
[E85]  321: /* Zeichensatz aus 'datei' laden */
[P7Ø]  322: {
[7S7]  323:   char d_name[MAXPATH],              /* gewäh
       lter Pfad */
[N28]  324:        f_name[MAXFILE],              /* gewäh
       lter Dateiname */
[489]  325:        f_ext[MAXEXT],                /* gewäh
       lte Dateierweiterung */
[W4A]  326:        titel[MAXFILE+MAXEXT+8];      /* enthä
       lt: "<Dateiname> laden:" */
[LCB]  327:   z_bytes z_buffer[96];              /* Zwisc
       henspeicher für Zeichensatz */
[6E6]  328:   int n,                            /* Schlei
       fenzähler */
[R76]  329:       knopf;                        /* gewähl
       ter Knopf */
[PZ8]  33Ø:   long  gewaehlt;                    /* wenn
       Zeichen gewählt := OK */
[SA9]  331:   form  format2;                     /* Zwisch
       enspeicher für Format */
[ØØØ]  332:
[U12]  333:   mouse_off();
[933]  334:   invertwahl(LADEN);
[V6D]  335:   strcpy(d_name,datei_box(413,128,"*.FNT","
       Zeichensatz laden","",BLACK,LIGHTMAGENTA));
[2P2]  336:   if (d_name[Ø]!=Ø)
[P7Ø]  337:   {
[427]  338:     fnsplit(d_name,NULL,NULL,f_name,f_ext);
[AP3]  339:     strcpy(titel,f_name);
[3G3]  34Ø:     strcat(titel,f_ext);
[ØM3]  341:     strcat(titel," laden:");
[625]  342:     knopf = box(413,128,639,236,titel,
[A48]  343:                 "Soll kompletter Zeichensat
       z|geladen werden ?",
[LA8]  344:                 "   (J)a    "," (N)ein   "
       ," (A)bbruch ",BLACK,LIGHTMAGENTA);
[F52]  345:     if (knopf == 2)
[P7Ø]  346:     {
[EHA]  347:       if (lese_datei(d_name,z_buffer,&forma
       t2) == ERROR) goto BEENDEN;
[NS6]  348:       zeichensatz_tabelle(z_buffer,format2)
       ;
[AK6]  349:       gewaehlt = auswahl("Zeichensatz laden
       ",
[XR8]  35Ø:                 "Bitte markieren Sie li
       nks|die zu ladenden Zeichen !",
[8A3]  351:                 BLACK,LIGHTMAGENTA);
[PR2]  352:       if (gewaehlt==OK)
```

Listing 1. »ze.c« enthält das Hauptprogramm des Zeichensatzeditors

```
[P7Ø]  353:        {
[U12]  354:          mouse off();
[8W2]  355:          for(n=Ø ; n<96 ; n++)
[T44]  356:            if (markiert[n]==Øxffff)
[P7Ø]  357:            {
[PE8]  358:              memcpy(zeichen[n].posxy,z_buffe
       r[n].posxy,111);
[ES4]  359:              format.dl[n] = format2.dl[n];
[U7Ø]  36Ø:            }
[PØ2]  361:          mouse_on();
[U7Ø]  362:        }
[8W2]  363:        for(n=Ø ; n<96 ; n++)
[T17]  364:          if (markiert[n]==Øxffff) invertwahl
       (n+32);
[G69]  365:        if ((datei[Ø]==Ø) && (gewaehlt!=NIX))
        strcpy(datei,d_name);
[327]  366:        if (gewaehlt == NIX)  box(413,128,639
       ,236,titel,
[636]  367:                                  "Es wurde k
       ein Zeichen|angewaehlt !",
[2X4]  368:                                  " (O)k ","
       ","",BLACK,LIGHTMAGENTA);
[ZRØ]  369:        else
[P7Ø]  37Ø:        {
[2R3]  371:          format.dØ = format2.dØ;
[CR3]  372:          format.d2 = format2.d2;
[U7Ø]  373:        }
[U7Ø]  374:      }
[C52]  375:      if (knopf == 1)
[P7Ø]  376:      {
[KY9]  377:        if (lese_datei(d_name,zeichen,&format
       ) == ERROR) goto BEENDEN;
[1N3]  378:        strcpy(datei,d_name);
[U7Ø]  379:      }
[222]  38Ø:      if (knopf != 3)
[1D6]  381:        zeichensatz_tabelle(zeichen,format);
[U7Ø]  382:    }
[YB1]  383: BEENDEN:
[933]  384:    invertwahl(LADEN);
[U7Ø]  385: }
[ØØØ]  386:
[4U6]  387: /*.....................................
       ..............*/
[ØØØ]  388:
[9N7]  389: int lese_datei(char *d_name,        /* Date
       iname */
[TE8]  39Ø:                z_bytes *z_buffer,     /* Spe
       icher für Zeichensatz */
[RU6]  391:                form *format2)         /* Spe
       icher für Format */
[ØØØ]  392:
[A44]  393: /* Zeichensatz einlesen */
[P7Ø]  394: {
[685]  395:   FILE *file;           /* Dateizeiger */
[K49]  396:   int z_bytes,         /* Anzahl der Daten f
       ür ein Zeichen */
[XT4]  397:       i;               /* Schleifenzähler */
[NA7]  398:   char art[14];         /* Speicher für Date
       ikopf */
[ØØØ]  399:
[Z34]  4ØØ:   file = fopen(d_name,"rb");
[6E2]  4Ø1:   if (file==NULL)
[P7Ø]  4Ø2:   {
[YR6]  4Ø3:     error_box("Fehler beim Lesen der Datei
       !");
[PD2]  4Ø4:     return(ERROR);
[U7Ø]  4Ø5:   }
[C32]  4Ø6:   mouse_uhr();
[V23]  4Ø7:   fread(art,14,1,file);
[T44]  4Ø8:   invertwahl(schriftart);
[AZ1]  4Ø9:   z_bytes = Ø;
[HM4]  41Ø:   if (strcmp(art,"DRAFT       ")==Ø)
[P7Ø]  411:   {
[432]  412:     z_bytes = 27;
[W93]  413:     schriftart = DRAFT;
[U7Ø]  414:   }
[HM4]  415:   if (strcmp(art,"ELITE       ")==Ø)
[P7Ø]  416:   {
[P32]  417:     z_bytes = 69;
[Y83]  418:     schriftart = ELITE;
[U7Ø]  419:   }
[9H4]  42Ø:   if (strcmp(art,"PICA        ")==Ø)
[P7Ø]  421:   {
[W32]  422:     z_bytes = 87;
[XZ2]  423:     schriftart = PICA;
[U7Ø]  424:   }
[8G5]  425:   if (strcmp(art,"PROPORTIONAL")==Ø)
[P7Ø]  426:   {
[R72]  427:     z_bytes = 111;
[4L4]  428:     schriftart = PROPORTIONAL;
[U7Ø]  429:   }
[T44]  43Ø:   invertwahl(schriftart);
[WG2]  431:   if (z_bytes != Ø)
[P7Ø]  432:   {
[FT3]  433:     fread(format2,98,1,file);
[2R5]  434:     CLS_ZEICHEN(z_buffer[Ø].posxy);
[VU2]  435:     for(i=Ø ; i<96 ; i++)
[L37]  436:       fread(z_buffer[i].posxy,z_bytes,1,fil
       e);
[F11]  437:   } else
[P7Ø]  438:   {
[VG2]  439:     mouse_pfeil();
[KMC]  44Ø:     error_box("Dieses Format ist nicht ZE-k
       ompatibel !|(Versuchen Sie Import)");
[S22]  441:     fclose(file);
[PD2]  442:     return(ERROR);
[U7Ø]  443:   }
[S22]  444:   fclose(file);
[VG2]  445:   mouse_pfeil();
[KW1]  446:   return(OK);
[U7Ø]  447: }
[ØØØ]  448:
[4U6]  449: /*.....................................
       ..............*/
[ØØØ]  45Ø:
[ZZ2]  451: void import(void)
[ØØØ]  452:
[CR5]  453: /* fremde Zeichensätze einlesen */
[P7Ø]  454: {
[7S7]  455:   char d_name[MAXPATH],             /* gewäh
       lter Pfad */
[N28]  456:        f_name[MAXFILE],             /* gewäh
       lter Dateiname */
[489]  457:        f_ext[MAXEXT],               /* gewäh
       lte Dateierweiterung */
[W4A]  458:        titel[MAXFILE+MAXEXT+8];     /* enthä
       lt: "<Dateiname> laden:" */
[LCB]  459:   z_bytes z_buffer[96];             /* Zwisc
       henspeicher für Zeichensatz */
[6E6]  46Ø:   int n,                           /* Schlei
       fenzähler */
[EF8]  461:       breite,                      /* Breit
       e eines Zeichens in Pixel */
[R76]  462:       knopf;                       /* gewähl
       ter Knopf */
[PZ8]  463:   long  gewaehlt;                  /* wenn
       Zeichen gewählt := OK */
[ØØØ]  464:
[U12]  465:   mouse_off();
[UE3]  466:   invertwahl(IMPORT);
[433]  467:   switch(schriftart)
[P7Ø]  468:   {
[DU3]  469:     case DRAFT:  breite = 9;
[PØ1]  47Ø:                  break;
[1W3]  471:     case ELITE:  breite = 23;
[PØ1]  472:                  break;
[1K3]  473:     case PICA:   breite = 29;
[PØ1]  474:                  break;
[355]  475:     case PROPORTIONAL:  breite = 37;
[PØ1]  476:                  break;
[U7Ø]  477:   }
[PØ2]  478:   mouse_on();
[ØX8]  479:   if (box(413,128,639,25Ø,"Import","Bitte w
       aehlen Sie aus:",
[1W8]  48Ø:          " (P)D-Softfont ","   (A)bbruch
        ","",BLACK,LIGHTMAGENTA) == 1)
[P7Ø]  481:   {
[13B]  482:     strcpy(d_name,datei_box(413,128,"*.*","
       Import","",BLACK,LIGHTMAGENTA));
[2P2]  483:     if (d_name[Ø]!=Ø)
[P7Ø]  484:     {
[427]  485:       fnsplit(d_name,NULL,NULL,f_name,f_ext
       );
[AP3]  486:       strcpy(titel,f_name);
[3G3]  487:       strcat(titel,f_ext);
[ØM3]  488:       strcat(titel," laden:");
[625]  489:       knopf = box(413,128,639,236,titel,
[A48]  49Ø:                   "Soll kompletter Zeichens
       atz|geladen werden ?",
[LA8]  491:                   "   (J)a    ","  (N)ein
        "," (A)bbruch ",BLACK,LIGHTMAGENTA);
[F52]  492:       if (knopf == 2)
[P7Ø]  493:       {
[5H9]  494:         if (lese_import(d_name,z_buffer) ==
        ERROR) goto BEENDEN;
[ZL6]  495:         zeichensatz_tabelle(z_buffer,format
       );
[AL4]  496:         gewaehlt = auswahl("Import",
[XR8]  497:                            "Bitte markieren Sie
       links|die zu ladenden Zeichen !",
[8A3]  498:                            BLACK,LIGHTMAGENTA);
[PR2]  499:         if (gewaehlt==OK)
[P7Ø]  5ØØ:         {
[U12]  5Ø1:           mouse_off();
[8W2]  5Ø2:           for(n=Ø ; n<96 ; n++)
[T44]  5Ø3:             if (markiert[n]==Øxffff)
[P7Ø]  5Ø4:             {
[PE8]  5Ø5:               memcpy(zeichen[n].posxy,z_buf
       fer[n].posxy,111);
[FL3]  5Ø6:               format.dl[n] = breite;
[U7Ø]  5Ø7:             }
[PØ2]  5Ø8:           mouse_on();
[U7Ø]  5Ø9:         }
[8W2]  51Ø:         for(n=Ø ; n<96 ; n++)
[T17]  511:           if (markiert[n]==Øxffff) invertwa
       hl(n+32);
```

```
[G69]  512:          if ((datei[Ø]==Ø) && (gewaehlt!=NIX
       )) strcpy(datei,d_name);
[327]  513:          if (gewaehlt == NIX)  box(413,128,6
       39,236,titel,
[636]  514:                                "Es wurde
        kein Zeichen|angewaehlt !",
[2X4]  515:                                " (O)k ",
       "","",BLACK,LIGHTMAGENTA);
[U7Ø]  516:          }
[C52]  517:          if (knopf == 1)
[P7Ø]  518:          {
[P39]  519:            if (lese_import(d_name,zeichen) ==
       ERROR) goto BEENDEN;
[1N3]  52Ø:            strcpy(datei,d_name);
[AP2]  521:            reset_format();
[U7Ø]  522:          }
[222]  523:          if (knopf != 3)
[1D6]  524:            zeichensatz_tabelle(zeichen,format)
       ;
[U7Ø]  525:       }
[U7Ø]  526:     }
[YB1]  527: BEENDEN:
[UE3]  528:   invertwahl(IMPORT);
[U7Ø]  529: }
[ØØØ]  53Ø:
[4U6]  531: /*.....................................
       ...............*/
[ØØØ]  532:
[TØ8]  533: int lese_import(char *d_name,           /* D
       ateiname */
[GE8]  534:                  z_bytes *z_buffer)    /* Sp
       eicher für Zeichensatz */
[ØØØ]  535:
[VD5]  536: /* fremder Zeichensatz einlesen */
[P7Ø]  537: {
[685]  538:   FILE *file;           /* Dateizeiger */
[9H5]  539:   int i;               /* Schleifenzähler */
[ØØØ]  54Ø:
[Z34]  541:   file = fopen(d_name,"rb");
[6E2]  542:   if (file==NULL)
[P7Ø]  543:   {
[YR6]  544:     error_box("Fehler beim Lesen der Datei
       !");
[PD2]  545:     return(ERROR);
[U7Ø]  546:   }
[C32]  547:   mouse_uhr();
[2R5]  548:   CLS_ZEICHEN(z_buffer[Ø].posxy);
[ZU2]  549:   for(i=1 ; i<96 ; i++)
[U48]  55Ø:     fread(z_buffer[i].posxy,format.dl[i]*3,
       1,file);
[S22]  551:   fclose(file);
[VG2]  552:   mouse_pfeil();
[KW1]  553:   return(OK);
[U7Ø]  554: }
[ØØØ]  555:
[4U6]  556: /*.....................................
       ...............*/
[ØØØ]  557:
[YW4]  558: void zeichen_speichern(void)
[ØØØ]  559:
[ZH4]  56Ø: /* Zeichensatz abspeichern */
[P7Ø]  561: {
[5F7]  562:   char d_name[MAXPATH];        /* gewählter
        Pfad */
[K8B]  563:   struct ffblk ff_blk;        /* Informatio
       nen über die zu ladende Datei */
[ØØØ]  564:
[U12]  565:   mouse_off();
[BW3]  566:   invertwahl(SPEICHERN);
[PØ2]  567:   mouse_on();
[2AB]  568:   strcpy(d_name,datei_box(413,128,"*.FNT","
       Zeichensatz speichern",datei,
[CD3]  569:              BLACK,LIGHTMAGENTA));
[YV2]  57Ø:   if (d_name[Ø] != Ø)
[P7Ø]  571:   {
[VL5]  572:     if (findfirst(d_name,&ff_blk,32)==Ø)
[P7Ø]  573:     {
[T97]  574:       if (box(413,128,639,22Ø,"Zeichensatz
       speichern",
[DU7]  575:               "Datei schon vorhanden !|Datei
       ueberschreiben ?",
[NM2]  576:               "  (J)a  "," (N)ein ",
[Ø34]  577:               "",BLACK,LIGHTMAGENTA) == 1)
[P7Ø]  578:       {
[S64]  579:         speichere_datei(d_name);
[1N3]  58Ø:         strcpy(datei,d_name);
[U7Ø]  581:       }
[F11]  582:     } else
[P7Ø]  583:     {
[S64]  584:       speichere_datei(d_name);
[1N3]  585:       strcpy(datei,d_name);
[U7Ø]  586:     }
[U7Ø]  587:   }
[U12]  588:   mouse_off();
[BW3]  589:   invertwahl(SPEICHERN);
[PØ2]  59Ø:   mouse_on();
[U7Ø]  591: }
[ØØØ]  592:
```

```
[4U6]  593: /*.....................................
       ...............*/
[ØØØ]  594:
[718]  595: void speichere_datei(char *datei)    /* Dat
       einame */
[ØØØ]  596:
[9ØB]  597: /* aktueller Zeichensatz (Variabeln: zeiche
       n,format) im Pfad datei
[PA2]  598:    abspeichern */
[P7Ø]  599: {
[SW4]  6ØØ:   FILE *file;          /* Dateizeiger */
[KA8]  6Ø1:   int k,f,z,e,        /* Anzahl der geschrieb
       enen Bytes */
[SP4]  6Ø2:       i,              /* Schleifenzähler */
[RH7]  6Ø3:       z_bytes;        /* Datengröße eines Zei
       chens */
[ØØØ]  6Ø4:
[C32]  6Ø5:   mouse_uhr();
[FP3]  6Ø6:   file = fopen(datei,"wb");
[6E2]  6Ø7:   if (file==NULL)
[SV6]  6Ø8:     error_box("Fehler beim Oeffnen der Date
       i !");
[ZRØ]  6Ø9:   else
[P7Ø]  61Ø:   {
[433]  611:     switch(schriftart)
[P7Ø]  612:     {
[857]  613:       case DRAFT:  k = fwrite("DRAFT
       ",13,1,file);
[432]  614:                    z_bytes = 27;
[PØ1]  615:                    break;
[C37]  616:       case ELITE:  k = fwrite("ELITE
       ",13,1,file);
[P32]  617:                    z_bytes = 69;
[PØ1]  618:                    break;
[RN6]  619:       case PICA:   k = fwrite("PICA        "
       ,13,1,file);
[W32]  62Ø:                    z_bytes = 87;
[PØ1]  621:                    break;
[883]  622:       case PROPORTIONAL:
[NU5]  623:                    k = fwrite("PROPORTIONAL"
       ,13,1,file);
[R72]  624:                    z_bytes = 111;
[PØ1]  625:                    break;
[U7Ø]  626:     }
[9X2]  627:     e = fputc(26,file);
[RM4]  628:     f = fwrite(&format,98,1,file);
[VTØ]  629:     z = 1;
[ET3]  63Ø:     if ((k==1) && (f==1) && (e==26))
[P7Ø]  631:     {
[VU2]  632:       for(i=Ø ; i<96 ; i++)
[P7Ø]  633:       {
[WK7]  634:         z = fwrite(zeichen[i].posxy,z_bytes
       ,1,file);
[882]  635:         if (z!=1) break;
[U7Ø]  636:       }
[U7Ø]  637:     }
[SN5]  638:     if ((k!=1) || (f!=1) || (z!=1) || (e!=2
       6))
[5E7]  639:       error_box("Fehler beim Schreiben der
       Datei !");
[S22]  64Ø:     fclose(file);
[U7Ø]  641:   }
[VG2]  642:   mouse_pfeil();
[U7Ø]  643: }
[ØØØ]  644:
[4U6]  645: /*.....................................
       ...............*/
[ØØØ]  646:
[H23]  647: void drucker(void)
[ØØØ]  648:
[6X4]  649: /* Zeichensatz installieren */
[P7Ø]  65Ø: {
[ZE7]  651:   void *picptr;             /* Bildzwischens
       peicher */
[XH5]  652:   int x,y,                 /* Mausposition *
       /
[DE4]  653:       taste,               /* Maustaste */
[A85]  654:       n,i,                 /* Schleifenzähle
       r */
[G54]  655:       key;                 /* Tastatur */
[C86]  656:   long wahl;               /* gewählter Knop
       f */
[ØØØ]  657:
[U12]  658:   mouse_off();
[7G3]  659:   invertwahl(DRUCKER);
[6Ø7]  66Ø:   picptr = getimage_in_puffer(413,1Ø8,639,3
       4Ø);
[98B]  661:   putbox(413,1Ø8,639,34Ø,"Zeichensatz insta
       llieren","",BLACK,LIGHTGREEN);
[5R2]  662:   setcolor(BLACK);
[5Y2]  663:   line(416,159,636,159);
[GH5]  664:   outtextxy(526,137,"installieren");
[6P5]  665:   set_knopf(422,127,518,139,"k(o)mplett");
[YU5]  666:   set_knopf(422,143,518,155,"t(e)ilweise");
[5R2]  667:   setcolor(BLACK);
[ZW2]  668:   line(416,3Ø4,636,3Ø4);
```

Listing 1. »ze.c« enthält das Hauptprogramm des Zeichensatzeditors

```
[D75]  669:   outtextxy(430,165,"Zeichensatz");
[433]  670:   switch(schriftart)
[P70]  671:   {
[SC6]  672:     case DRAFT:  outtextxy(526,165,"DRAFT:"
               );
[P01]  673:                  break;
[UB6]  674:     case ELITE:  outtextxy(526,165,"ELITE:"
               );
[P01]  675:                  break;
[0Z5]  676:     case PICA:   outtextxy(526,165,"PICA:");
[P01]  677:                  break;
[A59]  678:     case PROPORTIONAL:  outtextxy(526,165,"
               PROPORTIONAL:");
[P01]  679:                  break;
[U70]  680:   }
[147]  681:   set_knopf(453,177,600,189,"(U)mrahmte Sch
               rift");
[L47]  682:   set_knopf(453,193,600,205,"(S)chattenschr
               ift");
[VW5]  683:   set_knopf(453,209,600,221,"(F)ettdruck");
[9K5]  684:   set_knopf(453,225,600,237,"(K)ursiv");
[866]  685:   set_knopf(453,241,600,253,"(B)reitdruck")
               ;
[MW5]  686:   set_knopf(453,257,600,269,"Ho(c)hdruck");
[FH6]  687:   set_knopf(453,273,600,285,"(H)ochstellung
               ");
[EH6]  688:   set_knopf(453,289,600,301,"(T)iefstellung
               ");
[EF6]  689:   set_knopf(453,307,600,319,"(I)nstallieren
               ");
[XL5]  690:   set_knopf(453,323,600,335,"(A)bbruch");
[Y65]  691:   if (schriftart == PROPORTIONAL)
[P70]  692:   {
[5R2]  693:     setcolor(BLACK);
[003]  694:     line(475,279,577,279);
[SZ2]  695:     line(475,281,577,281);
[803]  696:     line(475,295,577,295);
[C03]  697:     line(475,297,577,297);
[HDA]  698:     if (drucker_mode & HOCHSTELLUNG) drucke
               r_mode -= HOCHSTELLUNG;
[TDA]  699:     if (drucker_mode & TIEFSTELLUNG) drucke
               r_mode -= TIEFSTELLUNG;
[U70]  700:   }
[FL6]  701:   invert_drucker_taste(drucker_mode);
[P02]  702:   mouse_on();
[NB0]  703:   do
[P70]  704:   {
[U71]  705:     wahl = 0;
[NB0]  706:     do
[P70]  707:     {
[V24]  708:       mouse_info(&x,&y,&taste);
[FY2]  709:     } while(taste!=0);
[NB0]  710:     do
[P70]  711:     {
[V24]  712:       mouse_info(&x,&y,&taste);
[YN1]  713:       if (kbhit())
[P70]  714:       {
[BR4]  715:         key = toupper(get_taste());
[5X1]  716:         switch(key)
[P70]  717:         {
[7Z3]  718:           case 'U': wahl = OUTLINE;
[P01]  719:                     break;
[AU3]  720:           case 'S': wahl = SHADOW;
[P01]  721:                     break;
[344]  722:           case 'O': wahl = KOMPLETT;
[P01]  723:                     break;
[0B4]  724:           case 'E': wahl = TEILWEISE;
[P01]  725:                     break;
[EA4]  726:           case 'F': wahl = FETTDRUCK;
[P01]  727:                     break;
[ZU3]  728:           case 'K': wahl = KURSIV;
[P01]  729:                     break;
[GD4]  730:           case 'B':  wahl = BREITDRUCK;
[P01]  731:                     break;
[U74]  732:           case 'C':  wahl = HOCHDRUCK;
[P01]  733:                     break;
[AW4]  734:           case 'H': wahl = HOCHSTELLUNG;
[P01]  735:                     break;
[4Y4]  736:           case 'T': wahl = TIEFSTELLUNG;
[P01]  737:                     break;
[4U4]  738:           case 'I': wahl = INSTALLIEREN;
[P01]  739:                     break;
[PS3]  740:           case 'A': wahl = ABBRUCH;
[P01]  741:                     break;
[U70]  742:         }
[U70]  743:       }
[LT4]  744:     } while((taste==0) && (wahl==0));
[FY1]  745:     if (wahl == 0)
[Z45]  746:       wahl = get_drucker_taste(x,y);
[B12]  747:     switch(wahl)
[P70]  748:     {
[GG7]  749:       case KOMPLETT:  if (!(drucker_mode &
               KOMPLETT))
[P70]  750:                       {
[8W5]  751:                         invert_drucker_tast
               e(KOMPLETT);
[AY5]  752:                         invert_drucker_tast
               e(TEILWEISE);
```

```
[DK4]  753:                         drucker_mode |= KOM
               PLETT;
[EF4]  754:                         drucker_mode -= TEI
               LWEISE;
[U70]  755:                       }
[P01]  756:                       break;
[718]  757:       case TEILWEISE:  if (!(drucker_mode &
                TEILWEISE))
[P70]  758:                       {
[8W5]  759:                         invert_drucker_tast
               e(KOMPLETT);
[AY5]  760:                         invert_drucker_tast
               e(TEILWEISE);
[0B4]  761:                         drucker_mode -= KOM
               PLETT;
[VP4]  762:                         drucker_mode |= TEI
               LWEISE;
[U70]  763:                       }
[P01]  764:                       break;
[BY6]  765:       case OUTLINE:   if (drucker_mode & OU
               TLINE)
[W64]  766:                         drucker_mode -= OUT
               LINE;
[ZR0]  767:                       else
[9E4]  768:                         drucker_mode |= OUT
               LINE;
[BN5]  769:                       invert_drucker_taste(
               OUTLINE);
[P01]  770:                       break;
[8H6]  771:       case SHADOW:    if (drucker_mode & SH
               ADOW)
[D14]  772:                         drucker_mode -= SHA
               DOW;
[ZR0]  773:                       else
[U84]  774:                         drucker_mode |= SHA
               DOW;
[DF5]  775:                       invert_drucker_taste(
               SHADOW);
[P01]  776:                       break;
[VD7]  777:       case FETTDRUCK: if (drucker_mode & FE
               TTDRUCK)
[LH4]  778:                         drucker_mode -= FET
               TDRUCK;
[ZR0]  779:                       else
[0T4]  780:                         drucker_mode |= FET
               TDRUCK;
[MX5]  781:                       invert_drucker_taste(
               FETTDRUCK);
[P01]  782:                       break;
[PP6]  783:       case KURSIV:    if (drucker_mode & KU
               RSIV)
[844]  784:                         drucker_mode -= KUR
               SIV;
[ZR0]  785:                       else
[NB4]  786:                         drucker_mode |= KUR
               SIV;
[2H5]  787:                       invert_drucker_taste(
               KURSIV);
[P01]  788:                       break;
[4P7]  789:       case BREITDRUCK:if (drucker_mode & BR
               EITDRUCK)
[LS4]  790:                         drucker_mode -= BRE
               ITDRUCK;
[ZR0]  791:                       else
[005]  792:                         drucker_mode |= BRE
               ITDRUCK;
[456]  793:                       invert_drucker_taste(
               BREITDRUCK);
[P01]  794:                       break;
[GL7]  795:       case HOCHDRUCK:  if (drucker_mode & H
               OCHDRUCK)
[5G4]  796:                         drucker_mode -= HOC
               HDRUCK;
[ZR0]  797:                       else
[KR4]  798:                         drucker_mode |= HOC
               HDRUCK;
[TX5]  799:                       invert_drucker_taste(
               HOCHDRUCK);
[P01]  800:                       break;
[S48]  801:       case HOCHSTELLUNG: if (schriftart!=PR
               OPORTIONAL)
[P70]  802:                       {
[995]  803:                         if (drucker_mode &
               HOCHSTELLUNG)
[R15]  804:                           drucker_mode -= H
               OCHSTELLUNG;
[ZR0]  805:                         else
[495]  806:                           drucker_mode |= H
               OCHSTELLUNG;
[795]  807:                         if (drucker_mode &
               TIEFSTELLUNG)
[P70]  808:                         {
[325]  809:                           drucker_mode -= T
               IEFSTELLUNG;
[NN6]  810:                           invert_drucker_ta
               ste(TIEFSTELLUNG);
[U70]  811:                         }
```

```
[BN6]  812:                              invert_drucker_tast
e(HOCHSTELLUNG);
[U7Ø]  813:                            }
[PØ1]  814:                            break;
[258]  815:       case TIEFSTELLUNG: if (schriftart!=PR
OPORTIONAL)
[P7Ø]  816:                            {
[795]  817:                              if (drucker_mode &
TIEFSTELLUNG)
[325]  818:                                drucker_mode -= T
IEFSTELLUNG;
[ZRØ]  819:                              else
[G95]  82Ø:                                drucker_mode |= T
IEFSTELLUNG;
[995]  821:                              if (drucker_mode &
HOCHSTELLUNG)
[P7Ø]  822:                              {
[R15]  823:                                drucker_mode -= H
OCHSTELLUNG;
[BN6]  824:                                invert_drucker_ta
ste(HOCHSTELLUNG);
[U7Ø]  825:                              }
[NN6]  826:                              invert_drucker_tast
e(TIEFSTELLUNG);
[U7Ø]  827:                            }
[PØ1]  828:                            break;
[ØX9]  829:       case INSTALLIEREN:  invert_drucker_ta
ste(INSTALLIEREN);
[PØ1]  83Ø:                            break;
[U7Ø]  831:     }
[BT7]  832:   } while((wahl!=INSTALLIEREN) && (wahl!=AB
BRUCH));
[U12]  833:   mouse_off();
[ØR5]  834:   putimage(413,1Ø8,picptr,COPY_PUT);
[ØR2]  835:   farfree(picptr);
[PØ2]  836:   mouse_on();
[4Z6]  837:   picptr = getimage_in_puffer(413,14Ø,639,2
ØØ);
[FV3]  838:   if (wahl==INSTALLIEREN)
[P7Ø]  839:   {
[KW7]  84Ø:     if (box(413,14Ø,639,24Ø,"Zeichensatz in
stallieren",
[ZL8]  841:             "Ist der Drucker|empfangsbereit ?
","  (J)a  "," (N)ein ","",
[EL3]  842:             BLACK,LIGHTGREEN) == 1)
[P7Ø]  843:     {
[U12]  844:       mouse_off();
[K39]  845:       putbox(413,14Ø,639,2ØØ,"","Drucker au
f ON LINE ?",BLACK,GREEN);
[C32]  846:       mouse_uhr();
[PØ2]  847:       mouse_on();
[CP2]  848:       drucker_init();
[916]  849:       if (druck_err == ERROR) goto BEENDEN;
[U12]  85Ø:       mouse_off();
[K59]  851:       putbox(413,14Ø,639,2ØØ,"","Zeichensat
z wird|installiert...",
[RØ3]  852:              BLACK,LIGHTGREEN);
[PØ2]  853:       mouse_on();
[P82]  854:       rom_in_ram();
[916]  855:       if (druck_err == ERROR) goto BEENDEN;
[8U3]  856:       if (schriftart == DRAFT)
[P7Ø]  857:       {
[652]  858:         draft_mode();
[916]  859:         if (druck_err == ERROR) goto BEENDE
N;
[U7Ø]  86Ø:       }
[TK2]  861:       else lq_mode();
[916]  862:       if (druck_err == ERROR) goto BEENDEN;
[N56]  863:       if (schriftart == ELITE) elite_mode()
;
[LW5]  864:       if (schriftart == PICA)   pica_mode()
;
[Y65]  865:       if (schriftart == PROPORTIONAL)
[P7Ø]  866:       {
[812]  867:         prop_mode();
[U7Ø]  868:       }
[CF3]  869:       else no_prop_mode();
[916]  87Ø:       if (druck_err == ERROR) goto BEENDEN;
[A99]  871:       if ((drucker_mode & (OUTLINE+SHADOW))
== OUTLINE+SHADOW)
[P7Ø]  872:       {
[3Ø4]  873:         outline_shadow_mode();
[U7Ø]  874:       }
[ZRØ]  875:       else
[SD4]  876:         if (drucker_mode & OUTLINE)
[P7Ø]  877:         {
[VN2]  878:           outline_mode();
[U7Ø]  879:         }
[ZRØ]  88Ø:         else
[934]  881:           if (drucker_mode & SHADOW)
[P7Ø]  882:           {
[VE2]  883:             shadow_mode();
[U7Ø]  884:           }
[ZRØ]  885:           else
[3K4]  886:             no_outline_shadow_mode();
[916]  887:       if (druck_err == ERROR) goto BEENDEN;
[7R4]  888:       if (drucker_mode & FETTDRUCK)
[P7Ø]  889:       {
[5Ø2]  89Ø:         fett_mode();
```

```
[U7Ø]  891:       }
[9E3]  892:       else no_fett_mode();
[916]  893:       if (druck_err == ERROR) goto BEENDEN;
[274]  894:       if (drucker_mode & KURSIV)
[P7Ø]  895:       {
[PH2]  896:         kursiv_mode();
[U7Ø]  897:       }
[UZ3]  898:       else no_kursiv_mode();
[916]  899:       if (druck_err == ERROR) goto BEENDEN;
[EW4]  9ØØ:       if (drucker_mode & BREITDRUCK)
[P7Ø]  9Ø1:       {
[163]  9Ø2:         breitdruck_mode();
[U7Ø]  9Ø3:       }
[5M4]  9Ø4:       else no_breitdruck_mode();
[916]  9Ø5:       if (druck_err == ERROR) goto BEENDEN;
[UM4]  9Ø6:       if (drucker_mode & HOCHDRUCK)
[P7Ø]  9Ø7:       {
[KX2]  9Ø8:         hochdruck_mode();
[U7Ø]  9Ø9:       }
[PB4]  91Ø:       else no_hochdruck_mode();
[916]  911:       if (druck_err == ERROR) goto BEENDEN;
[995]  912:       if (drucker_mode & HOCHSTELLUNG)
[P7Ø]  913:       {
[RY1]  914:         hoch_mode();
[U7Ø]  915:       }
[ZRØ]  916:       else
[P7Ø]  917:       {
[795]  918:         if (drucker_mode & TIEFSTELLUNG)
[P7Ø]  919:         {
[3Z1]  92Ø:           tief_mode();
[U7Ø]  921:         }
[UB4]  922:         else no_hoch_tief_mode();
[U7Ø]  923:       }
[916]  924:       if (druck_err == ERROR) goto BEENDEN;
[PM4]  925:       if (drucker_mode & KOMPLETT)
[P7Ø]  926:       {
[N54]  927:         zeichen_satz_install();
[916]  928:         if (druck_err == ERROR) goto BEENDE
N;
[6W2]  929:         benutz_z_satz();
[916]  93Ø:         if (druck_err == ERROR) goto BEENDE
N;
[U7Ø]  931:       }
[ZRØ]  932:       else
[P7Ø]  933:       {
[VG2]  934:         mouse_pfeil();
[A47]  935:         wahl = auswahl("Zeichensatz install
ieren",
[3P9]  936:                        "Bitte markieren Sie link
s|die Zeichen",BLACK,LIGHTGREEN);
[C32]  937:         mouse_uhr();
[R82]  938:         if (wahl == OK)
[P7Ø]  939:         {
[8W2]  94Ø:           for(n=Ø ; n<96 ; n++)
[P7Ø]  941:           {
[PT2]  942:             if (markiert[n])
[P7Ø]  943:             {
[ØM6]  944:               if (druck_err == OK) zeichen_
install(n);
[ES2]  945:               invertwahl(n+32);
[U7Ø]  946:             }
[U7Ø]  947:           }
[916]  948:           if (druck_err == ERROR) goto BEEN
DEN;
[6W2]  949:           benutz_z_satz();
[916]  95Ø:           if (druck_err == ERROR) goto BEEN
DEN;
[F11]  951:         } else
[P7Ø]  952:         {
[VG2]  953:           mouse_pfeil();
[8W2]  954:           for(n=Ø ; n<96 ; n++)
[PP5]  955:             if (markiert[n]) invertwahl(n+3
2);
[UP9]  956:           if (wahl == NIX) box(413,14Ø,639,
24Ø,"Zeichensatz installieren",
[636]  957:                                "Es wurde k
ein Zeichen|angewaehlt !",
[TF4]  958:                                " (O)k ","",
"",BLACK,LIGHTGREEN);
[U7Ø]  959:         }
[U7Ø]  96Ø:       }
[VG2]  961:       mouse_pfeil();
[K35]  962:       if ((wahl!=ABBRUCH) && (wahl!=NIX))
[P7Ø]  963:       {
[KW7]  964:         if (box(413,14Ø,639,24Ø,"Zeichensat
z installieren",
[YU8]  965:                 "Wollen Sie einen|Probeausd
ruck ?","  (J)a  "," (N)ein ","",
[MC3]  966:                 BLACK,LIGHTGREEN)==1)
[P7Ø]  967:         {
[RSØ]  968:           i = Ø;
[GG3]  969:           fprintf(stdprn,"\n");
[916]  97Ø:           if (druck_err == ERROR) goto BEEN
DEN;
[663]  971:           for(n=' ' ; n<128 ; n++)
[P7Ø]  972:           {
```

Listing 1. »ze.c« enthält das Hauptprogramm des Zeichensatzeditors

```
[NR2]  973:              fputc(n,stdprn);
[916]  974:              if (druck_err == ERROR) goto BE
       ENDEN;
[UFØ]  975:              i++;
[MF8]  976:              if (((i==4Ø) && (drucker_mode &
        BREITDRUCK)) || (i==8Ø))
[P7Ø]  977:              {
[UM4]  978:                if (drucker_mode & HOCHDRUCK)
[GG3]  979:                  fprintf(stdprn,"\n");
[916]  98Ø:                if (druck_err == ERROR) goto
       BEENDEN;
[GG3]  981:                fprintf(stdprn,"\n");
[916]  982:                if (druck_err == ERROR) goto
       BEENDEN;
[RSØ]  983:                i = Ø;
[U7Ø]  984:              }
[U7Ø]  985:            }
[UM4]  986:            if (drucker_mode & HOCHDRUCK)
[GG3]  987:              fprintf(stdprn,"\n");
[916]  988:            if (druck_err == ERROR) goto BEEN
       DEN;
[GG3]  989:            fprintf(stdprn,"\n");
[916]  99Ø:            if (druck_err == ERROR) goto BEEN
       DEN;
[U7Ø]  991:          }

[U7Ø]  992:        }
[U14]  993: BEENDEN: druck_err = OK;
[U12]  994:        mouse_off();
[8R5]  995:        putimage(413,14Ø,picptr,COPY_PUT);
[ØR2]  996:        farfree(picptr);
[PØ2]  997:        mouse_on();
[U7Ø]  998:      }
[U7Ø]  999:    }
[7G3] 1ØØØ:    invertwahl(DRUCKER);
[U7Ø] 1ØØ1: }
[ØØØ] 1ØØ2:
[4U6] 1ØØ3: /*.....................................
       ................*/
[ØØØ] 1ØØ4:
[9L5] 1ØØ5: void zeichen_satz_install(void)
[ØØØ] 1ØØ6:
[YR6] 1ØØ7: /* kompletter Zeichensatz installieren */
[P7Ø] 1ØØ8: {
[9X4] 1ØØ9:   int i;       /* Schleifenzähler */
[ØØØ] 1Ø1Ø:
[VU2] 1Ø11:   for(i=Ø ; i<96 ; i++)
[P7Ø] 1Ø12:   {
[9S2] 1Ø13:     invertwahl(i+32);
[LD3] 1Ø14:     zeichen_install(i);
[9S2] 1Ø15:     invertwahl(i+32);
[6W4] 1Ø16:     if (druck_err == ERROR) break;
[U7Ø] 1Ø17:   }
[U7Ø] 1Ø18: }
[ØØØ] 1Ø19:
[4U6] 1Ø2Ø: /*.....................................
       ................*/
[ØØØ] 1Ø21:
[DR4] 1Ø22: void zeichen_install(int i)
[ØØØ] 1Ø23:
[UK6] 1Ø24: /* Zeichen Nr.i im Drucker installieren */
[P7Ø] 1Ø25: {
[LX4] 1Ø26:   int n;       /* Schleifenzähler */
[ØØØ] 1Ø27:
[9B6] 1Ø28:   if (druck_err == OK) fputc(ESC,stdprn);
[X46] 1Ø29:   if (druck_err == OK) fputc('&',stdprn);
[B16] 1Ø3Ø:   if (druck_err == OK) fputc(Ø,stdprn);
[4A6] 1Ø31:   if (druck_err == OK) fputc(i+32,stdprn);
[4A6] 1Ø32:   if (druck_err == OK) fputc(i+32,stdprn);
[WD7] 1Ø33:   if (druck_err == OK) fputc(format.dØ,stdp
       rn);
[BØ8] 1Ø34:   if (druck_err == OK) fputc(format.d1[i],s
       tdprn);
[YD7] 1Ø35:   if (druck_err == OK) fputc(format.d2,stdp
       rn);
[H73] 1Ø36:   if (druck_err == OK)
[P7Ø] 1Ø37:   {
[WS4] 1Ø38:     for(n=Ø ; n<format.d1[i] ; n++)
[P7Ø] 1Ø39:     {
[RP5] 1Ø4Ø:       if (!(drucker_mode & TIEFSTELLUNG))
[7U6] 1Ø41:         fputc(zeichen[i].posxy[n][Ø],stdprn
       );
[6W4] 1Ø42:       if (druck_err == ERROR) break;
[BU6] 1Ø43:       fputc(zeichen[i].posxy[n][1],stdprn);
[6W4] 1Ø44:       if (druck_err == ERROR) break;
[DP5] 1Ø45:       if (!(drucker_mode & HOCHSTELLUNG))
[FU6] 1Ø46:         fputc(zeichen[i].posxy[n][2],stdprn
       );
[6W4] 1Ø47:       if (druck_err == ERROR) break;
[U7Ø] 1Ø48:     }
[U7Ø] 1Ø49:   }
[U7Ø] 1Ø5Ø: }
[ØØØ] 1Ø51:
[4U6] 1Ø52: /*.....................................
       ................*/
[ØØØ] 1Ø53:
[T9C] 1Ø54: void invert_drucker_taste(unsigned taste)
            /* Zu invertierende Taste */
[ØØØ] 1Ø55:
```

```
[H37] 1Ø56: /* Invertiert Tasten aus dem Druckermenü */
[P7Ø] 1Ø57: {
[U12] 1Ø58:   mouse_off();
[4V7] 1Ø59:   if (taste & OUTLINE)      invertimage(452
       ,176,6Ø1,19Ø);
[1K7] 1Ø6Ø:   if (taste & SHADOW)       invertimage(45
       2,192,6Ø1,2Ø6);
[KW7] 1Ø61:   if (taste & FETTDRUCK)    invertimage(45
       2,2Ø8,6Ø1,222);
[1N7] 1Ø62:   if (taste & KURSIV)       invertimage(45
       2,224,6Ø1,238);
[WY7] 1Ø63:   if (taste & BREITDRUCK)   invertimage(45
       2,24Ø,6Ø1,254);
[DY7] 1Ø64:   if (taste & HOCHDRUCK)    invertimage(452
       ,256,6Ø1,27Ø);
[W68] 1Ø65:   if (taste & HOCHSTELLUNG) invertimage(45
       2,272,6Ø1,286);
[178] 1Ø66:   if (taste & TIEFSTELLUNG) invertimage(45
       2,288,6Ø1,3Ø2);
[K58] 1Ø67:   if (taste & INSTALLIEREN) invertimage(45
       2,3Ø6,6Ø1,32Ø);
[VT7] 1Ø68:   if (taste & KOMPLETT)     invertimage(42
       1,126,519,14Ø);
[RX7] 1Ø69:   if (taste & TEILWEISE)    invertimage(421
       ,142,519,156);
[PØ2] 1Ø7Ø:   mouse_on();
[U7Ø] 1Ø71: }
[ØØØ] 1Ø72:
[4U6] 1Ø73: /*.....................................
       ................*/
[ØØØ] 1Ø74:
[V4A] 1Ø75: unsigned get_drucker_taste(int x,int y)
      /* Mausposition */
[ØØØ] 1Ø76:
[YN6] 1Ø77: /* liefert die gewählte Taste zurück */
[P7Ø] 1Ø78: {
[8X2] 1Ø79:   if ((x>453) && (x<6ØØ))
[P7Ø] 1Ø8Ø:   {
[YP5] 1Ø81:     if ((y>177) && (y<189)) return(OUTLINE)
       ;
[CH5] 1Ø82:     if ((y>193) && (y<2Ø5)) return(SHADOW);
[PB6] 1Ø83:     if ((y>2Ø9) && (y<221))  return(FETTDRU
       CK);
[5N5] 1Ø84:     if ((y>225) && (y<237)) return(KURSIV);
[3A6] 1Ø85:     if ((y>241) && (y<253)) return(BREITDRU
       CK);
[N16] 1Ø86:     if ((y>257) && (y<269)) return(HOCHDRUC
       K);
[VL6] 1Ø87:     if ((y>273) && (y<285)) return(HOCHSTEL
       LUNG);
[5M6] 1Ø88:     if ((y>289) && (y<3Ø3)) return(TIEFSTEL
       LUNG);
[LU6] 1Ø89:     if ((y>3Ø7) && (y<319))  return(INSTALL
       IEREN);
[KL5] 1Ø9Ø:     if ((y>323) && (y<335)) return(ABBRUCH)
       ;
[U7Ø] 1Ø91:   }
[CX2] 1Ø92:   if ((x>422) && (x<518))
[P7Ø] 1Ø93:   {
[8Z5] 1Ø94:     if ((y>127) && (y<139)) return(KOMPLETT
       );
[226] 1Ø95:     if ((y>143) && (y<155)) return(TEILWEIS
       E);
[U7Ø] 1Ø96:   }
[TL1] 1Ø97:   return(Ø);
[U7Ø] 1Ø98: }
[ØØØ] 1Ø99:
[4U6] 11ØØ: /*.....................................
       ................*/
[ØØØ] 11Ø1:
[K69] 11Ø2: int auswahl(char *titel,
            /* Überschrift        */
[1M7] 11Ø3:             char *text,
            /* auszugebender Text  */
[YE7] 11Ø4:             int color,
             /* Vordergrundfarbe    */
[WT7] 11Ø5:             int h_color)
             /* Hintergrundfarbe    */
[ØØØ] 11Ø6:
[NR8] 11Ø7: /* alle Zeichen können mit der Maus markier
       t werden,
[K69] 11Ø8:    Variable markiert[n] != Ø ==> Zeichen n
       wurde markiert */
[P7Ø] 11Ø9: {
[6G5] 111Ø:   int n,                /* Schleifenzähler */
[1V3] 1111:       key,              /* Tastatur */
[D44] 1112:       taste,            /* Maustaste */
[ØH4] 1113:       x,y;              /* Mausposition */
[WD6] 1114:   long  gewaehlt;        /* gewählter Knopf *
       /
[Z47] 1115:   void *picptr;          /* Bildzwischenspeic
       her */
[ØØØ] 1116:
[U12] 1117:   mouse_off();
[R17] 1118:   picptr = getimage_in_puffer(4Ø9,178,639,2
       7Ø);
[8W2] 1119:   for(n=Ø ; n<96 ; n++)
[PR2] 112Ø:     markiert[n] = Ø;
```

```
[AT7] 1121:   putbox(409,178,639,270,titel,text,color,h
        _color);
[VU5] 1122:   set_knopf(428,240,514,252,"Ok [Enter]");
[GB6] 1123:   set_knopf(518,240,628,252,"Abbruch [ESC]"
        );
[P02] 1124:   mouse_on();
[NB0] 1125:   do
[P70] 1126:   {
[NB0] 1127:     do
[P70] 1128:     {
[XZ1] 1129:       gewaehlt = 0;
[V24] 1130:       mouse_info(&x,&y,&taste);
[YN1] 1131:       if (kbhit())
[P70] 1132:       {
[V23] 1133:         key = get_taste();
[DU3] 1134:         if ((key>=' ') && (key <= 127))
[P70] 1135:         {
[7M4] 1136:           markiert[key-32] ^= 0xffff;
[9T2] 1137:           invertwahl(key);
[U70] 1138:         }
[7T5] 1139:         if (key == ESC)       gewaehlt = ABB
        RUCH;
[865] 1140:         if (key == RETURN)  gewaehlt = OK;
[U70] 1141:       }
[NH5] 1142:     } while((taste==0) && (gewaehlt==0));
[X07] 1143:     if ((x>2) && (x<401) && (y>2) && (y<325
        ) && (taste==1))
[P70] 1144:     {
[7D4] 1145:       gewaehlt=(x-2)/50+(y-2)/27*8+32;
[3E5] 1146:       markiert[gewaehlt-32] ^= 0xffff;
[CR3] 1147:       invertwahl(gewaehlt);
[U70] 1148:     }
[VN4] 1149:     if ((y>240) && (y<252) && (taste==1))
[P70] 1150:     {
[T55] 1151:       if ((x>434) && (x<514)) gewaehlt = OK
        ;
[E56] 1152:       if ((x>518) && (x<598)) gewaehlt = AB
        BRUCH;
[U70] 1153:     }
[NB0] 1154:     do
[P70] 1155:     {
[V24] 1156:       mouse_info(&x,&y,&taste);
[FY2] 1157:     } while(taste!=0);
[9C7] 1158:   } while((gewaehlt!=OK) && (gewaehlt!=ABBR
        UCH));
[PR2] 1159:   if (gewaehlt==OK)
[P70] 1160:   {
[P92] 1161:     gewaehlt=NIX;
[8W2] 1162:     for(n=0 ; n<96 ; n++)
[U75] 1163:       if (markiert[n]) gewaehlt = OK;
[U70] 1164:   }
[U12] 1165:   mouse_off();
[YR5] 1166:   putimage(409,178,picptr,COPY_PUT);
[0R2] 1167:   farfree(picptr);
[P02] 1168:   mouse_on();
[8Y2] 1169:   return(gewaehlt);
[U70] 1170: }
[000] 1171:
[4U6] 1172: /*.....
        ...............*/
[000] 1173:
[E34] 1174: void zwischenraum(void)
[000] 1175:
[4F8] 1176: /* Freiraum zwischen einzelnen Zeichen eins
        tellen */
[P70] 1177: {
[Z47] 1178:   void *picptr;          /* Bildzwischenspeic
        her */
[X75] 1179:   int x,y,              /* Mausposition */
[D44] 1180:       taste,            /* Maustaste */
[1V3] 1181:       key,              /* Tastatur */
[NBB] 1182:       max_frei,         /* maximale Anzahl f
        reier Punkt links & rechts neben dem
[2M1] 1183:                           Zeichen */
[G59] 1184:       d0,d2;            /* Zwischenspeicher v
        on format.d0, format.d2 */
[B26] 1185:   long  wahl;           /* gewählter Knopf *
        /
[KZ5] 1186:   char text[4];         /* String von d0,d1
        */
[000] 1187:
[U12] 1188:   mouse_off();
[TG4] 1189:   invertwahl(ZWISCHENRAUM);
[DZ6] 1190:   picptr = getimage_in_puffer(413,120,639,2
        50);
[KAA] 1191:   putbox(413,120,639,250,"Zwischenraum eins
        tellen","",BLACK,YELLOW);
[8R4] 1192:   set_knopf(430,220,524,232,"(O)k");
[UL5] 1193:   set_knopf(528,220,622,232,"(A)bbruch");
[5R2] 1194:   setcolor(BLACK);
[433] 1195:   switch(schriftart)
[P70] 1196:   {
[W54] 1197:     case DRAFT:  max_frei = 3;
[P01] 1198:                  break;
[1Y1] 1199:     case ELITE:
[0V3] 1200:     case PICA:   max_frei = 7;
[P01] 1201:                  break;
[5E5] 1202:     case PROPORTIONAL:  max_frei = 5;
[P01] 1203:                  break;
```

Listing 1. »ze.c« enthält das Hauptprogramm des Zeichensatzeditors

```
[U70] 1204:   }
[KS7] 1205:   outtextxy(420,140,"Abstand zwischen einze
        lnen");
[257] 1206:   outtextxy(420,152,"Zeichen:  (max.:    Pu
        nkte)");
[XW4] 1207:   sprintf(text,"%02i",max_frei);
[ZW3] 1208:   outtextxy(556,152,text);
[S36] 1209:   outtextxy(446,174,"links:       rechts:")
        ;
[Y94] 1210:   set_knopf(446,188,460,200,"-");
[6A4] 1211:   set_knopf(488,188,502,200,"+");
[3A4] 1212:   set_knopf(550,188,564,200,"-");
[BA4] 1213:   set_knopf(592,188,606,200,"+");
[EV5] 1214:   setfillstyle(SOLID_FILL,YELLOW);
[5R2] 1215:   setcolor(BLACK);
[YT3] 1216:   rectangle(463,186,485,202);
[PU3] 1217:   rectangle(567,186,589,202);
[MA2] 1218:   d0 = format.d0;
[0B2] 1219:   d2 = format.d2;
[P02] 1220:   mouse_on();
[NB0] 1221:   do
[P70] 1222:   {
[U12] 1223:     mouse_off();
[KT2] 1224:     bar(464,187,484,201);
[PU2] 1225:     bar(568,187,588,201);
[PM3] 1226:     sprintf(text,"%02i",d0);
[0X3] 1227:     outtextxy(467,191,text);
[WM3] 1228:     sprintf(text,"%02i",d2);
[TW3] 1229:     outtextxy(571,191,text);
[P02] 1230:     mouse_on();
[NB0] 1231:     do
[P70] 1232:     {
[V24] 1233:       mouse_info(&x,&y,&taste);
[AG2] 1234:     }  while(taste);
[U71] 1235:     wahl = 0;
[NB0] 1236:     do
[P70] 1237:     {
[V24] 1238:       mouse_info(&x,&y,&taste);
[952] 1239:       if (taste == 1)
[P70] 1240:       {
[FX2] 1241:         if ((y>188) && (y<200))
[P70] 1242:         {
[UZ4] 1243:           if ((x>446) && (x<460)) wahl = L_
        MIN;
[785] 1244:           if ((x>488) && (x<502)) wahl = L_
        PLUS;
[XZ4] 1245:           if ((x>550) && (x<564)) wahl = R_
        MIN;
[A85] 1246:           if ((x>592) && (x<606)) wahl = R_
        PLUS;
[U70] 1247:         }
[LW2] 1248:         if ((y>220) && (y<232))
[P70] 1249:         {
[BD4] 1250:           if ((x>430) && (x<524)) wahl = OK
        ;
[YC5] 1251:           if ((x>528) && (x<622)) wahl = AB
        BRUCH;
[U70] 1252:         }
[U70] 1253:       }
[YN1] 1254:       if (kbhit())
[P70] 1255:       {
[BR4] 1256:         key = toupper(get_taste());
[EM3] 1257:         if (key == 'O') wahl = OK;
[EK4] 1258:         if (key == 'A') wahl = ABBRUCH;
[U70] 1259:       }
[XS2] 1260:     } while(wahl==0);
[B12] 1261:     switch(wahl)
[P70] 1262:     {
[3B4] 1263:       case L_MIN:  if (d0 > 0) d0--;
[P01] 1264:                    break;
[W56] 1265:       case L_PLUS:if (d0 + d2 < max_frei) d
        0++;
[P01] 1266:                    break;
[5C4] 1267:       case R_MIN:  if (d2 > 0) d2--;
[P01] 1268:                    break;
[M66] 1269:       case R_PLUS:if (d0 + d2 < max_frei) d
        2++;
[P01] 1270:                    break;
[U70] 1271:     }
[BY5] 1272:   } while((wahl!=OK) && (wahl!=ABBRUCH));
[R82] 1273:   if (wahl == OK)
[P70] 1274:   {
[0B2] 1275:     format.d0 = d0;
[AB2] 1276:     format.d2 = d2;
[U70] 1277:   }
[U12] 1278:   mouse_off();
[0R5] 1279:   putimage(413,120,picptr,COPY_PUT);
[0R2] 1280:   farfree(picptr);
[P02] 1281:   mouse_on();
[TG4] 1282:   invertwahl(ZWISCHENRAUM);
[U70] 1283: }
[000] 1284:
[4U6] 1285: /*.....
        ...............*/
[000] 1286:
[S53] 1287: void edit(int wahl)
```

```
[ØØØ] 1288:
[ØG3] 1289: /* Zeichen editieren */
[P7Ø] 129Ø: {
[U88] 1291:   void *picptr1,*picptr2;    /* Bildzwische
nspeicher */
[589] 1292:   z_bytes z_edit,            /* Zwischenspe
icher für Zeichen */
[H6A] 1293:           z_ablage,          /* Zwischenspe
icher für Ablage (für tauschen) */
[G19] 1294:           z_puffer;          /* Zwischenspe
icher für Zeichen drehen */
[X8C] 1295:   char z_ablage_form;        /* Zwischenspe
icher für Zeichenbreite in Ablage */
[FN5] 1296:   int x,y,                   /* Mausposition
 */
[SA5] 1297:       i,                     /* Schleifenzäh
ler */
[RF4] 1298:       taste,                 /* Maustaste */
[2Z9] 1299:       zeichnen,              /* zeichnen !=
 Ø => Punkt kann gesetzt werden */
[KY4] 13ØØ:       farbe,                 /* Zeichenfarbe
 */
[T74] 13Ø1:       key,                   /* Tastatur */
[NV6] 13Ø2:       x1,y1,                 /* Bitposition
vom Zeichen */
[EH8] 13Ø3:       x_abbr,                /* höchste X-P
osition von Zeichen */
[TB6] 13Ø4:       x_step,                /* Breite eine
s Pixels */
[H4A] 13Ø5:       spalten_puffer,        /* anfängliche
 Spaltenzahl festhalten */
[VC6] 13Ø6:       spalten,               /* Anzahl der S
palten */

[7N8] 13Ø7:       alt_spalten;           /* Anzahl der v
orherigen Spalten */
[N37] 13Ø8:   unsigned knopf;            /* gewählter K
nopf */
[D58] 13Ø9:   char text[4Ø],             /* String Zwisc
henspeicher */
[UØ6] 131Ø:       *puffer;               /* Zeiger auf P
ixel */
[ØØØ] 1311:
[648] 1312:   memcpy(z_edit.posxy,zeichen[wahl-32].posx
y,111);
[U12] 1313:   mouse_off();
[WT6] 1314:   picptr1 = getimage_in_puffer(1Ø,1Ø,636,17
5);
[8Ø7] 1315:   picptr2 = getimage_in_puffer(1Ø,176,636,3
4Ø);
[1C7] 1316:   putbox(1Ø,1Ø,636,34Ø,"- Editor -","",BLAC
K,YELLOW);
[5R2] 1317:   setcolor(BLACK);
[433] 1318:   switch(schriftart)
[P7Ø] 1319:   {
[CX3] 132Ø:     case DRAFT: x_abbr = 566;
[FW1] 1321:                 x_step = 45;
[TN2] 1322:                 ablage_form = 9;
[PØ1] 1323:                 break;
[G44] 1324:     case ELITE:  x_abbr = 552;
[EW1] 1325:                 x_step = 17;
[BS2] 1326:                 ablage_form = 23;
[PØ1] 1327:                 break;
[FR3] 1328:     case PICA:  x_abbr = 567;
[2W1] 1329:                 x_step = 14;
[HS2] 133Ø:                 ablage_form = 29;
[PØ1] 1331:                 break;
[T85] 1332:     case PROPORTIONAL:  x_abbr =568;
[PV1] 1333:                 x_step = 11;
[YP5] 1334:                 setfillstyle(SOLID_FILL,WHI
TE);
[8T2] 1335:                 bar(167,324,573,332);
[YT3] 1336:                 rectangle(166,323,574,333);
[ST5] 1337:                 outtextxy(5Ø,325,"Zeichenbr
eite:");
[PØ1] 1338:                 break;
[U7Ø] 1339:   }
[C6A] 134Ø:   spalten_puffer = spalten = alt_spalten =
format.dl[wahl-32];
[1EB] 1341:   set_raster_char(x_step,spalten,alt_spalte
n,(char *)z_edit.posxy);
[5R2] 1342:   setcolor(BLACK);
[UH5] 1343:   for(x=16Ø ; x<x_abbr ; x+=5*x_step)
[P7Ø] 1344:   {
[UG2] 1345:     line(x,3Ø5,x,31Ø);
[HF5] 1346:     sprintf(text,"%2i",(x-16Ø)/x_step);
[B14] 1347:     outtextxy(x-8,312,text);
[U7Ø] 1348:   }
[PE4] 1349:   outtextxy(57Ø,312,"Spalten");
[1C4] 135Ø:   outtextxy(57Ø,46,"Gross-");
[FZ3] 1351:   outtextxy(57Ø,55,"buch-");
[LA4] 1352:   outtextxy(57Ø,64,"staben");
[584] 1353:   outtextxy(57Ø,242,"Grund-");
[V64] 1354:   outtextxy(57Ø,251,"linie");
[Y44] 1355:   outtextxy(57Ø,131,"Tief-");
[LS4] 1356:   outtextxy(57Ø,14Ø,"stellung");
[MØ4] 1357:   outtextxy(57Ø,149,"   \31");
[PØ4] 1358:   outtextxy(57Ø,189,"   \3Ø");
[M54] 1359:   outtextxy(57Ø,198,"Hoch-");
[1T4] 136Ø:   outtextxy(57Ø,2Ø7,"stellung");
[L6B] 1361:   sprintf(text,"Zeichen '%c' (%2X-hex) wird
 editiert:",(char)wahl,wahl);
[8N3] 1362:   outtextxy(2Ø,3Ø,text);
[XF5] 1363:   set_knopf(21,126,1Ø5,138,"a(b)legen");
[815] 1364:   set_knopf(21,142,1Ø5,154,"(h)olen");
[MP5] 1365:   set_knopf(21,158,1Ø5,17Ø,"(t)auschen");
[TK5] 1366:   set_knopf(21,184,1Ø5,196,"(S)chwarz");
[7Ø5] 1367:   set_knopf(21,2ØØ,1Ø5,212,"(W)eiss");
[C65] 1368:   set_knopf(21,226,1Ø5,238,"(d)rehen");
[1U4] 1369:   set_knopf(21,252,1Ø5,264,"(N)eu");
[Ø85] 137Ø:   set_knopf(21,278,1Ø5,29Ø,"(F)ertig");
[3E5] 1371:   set_knopf(21,294,1Ø5,3Ø6,"(A)bbruch");
[2F4] 1372:   set_knopf(127,184,141,196,"\36");
[DE4] 1373:   set_knopf(127,2ØØ,141,212,"\37");
[2E4] 1374:   set_knopf(112,192,126,2Ø4,"\21");
[5E4] 1375:   set_knopf(142,192,156,2Ø4,"\2Ø");
[5R2] 1376:   setcolor(BLACK);
[ØX6] 1377:   settextstyle(DEFAULT_FONT,VERT_DIR,1);
[ØD5] 1378:   outtextxy(138,212,"verschieben:");
[C47] 1379:   settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
[ØB4] 138Ø:   outtextxy(38,88,"Ablage:");
[DB3] 1381:   rectangle(4Ø,96,8Ø,123);
[XA3] 1382:   rectangle(4Ø,38,8Ø,65);
[778] 1383:   print_zeichen(42,98,(char *)ablage.posxy,
spalten);
[SP2] 1384:   farbe = SCHWARZ;
[3Z1] 1385:   zeichnen = Ø;
[FG4] 1386:   invert_edit_taste(farbe);
[PØ2] 1387:   mouse_on();
[NBØ] 1388:   do
[P7Ø] 1389:   {
[YØ2] 139Ø:     knopf = NIX;
[V24] 1391:     mouse_info(&x,&y,&taste);
[KV9] 1392:     if ((x>16Ø) && (x<16Ø+x_step*spalten) &
& (y>4Ø) && (y<3Ø4) && (taste==1))
[X32] 1393:       zeichnen = -1;
[ZRØ] 1394:     else
[P7Ø] 1395:     {
[5CA] 1396:       if (zeichnen) print_zeichen(42,4Ø,(ch
ar *)z_edit.posxy,spalten);
[3Z1] 1397:       zeichnen = Ø;
[U7Ø] 1398:     }
[7C9] 1399:     if ((zeichnen==Ø) && (get_edit_taste(x,
y)!=NIX) && (taste==1))
[P7Ø] 14ØØ:     {
[NBØ] 14Ø1:       do
[P7Ø] 14Ø2:       {
[V24] 14Ø3:         mouse_info(&x,&y,&taste);
[413] 14Ø4:       } while(taste==1);
[8N4] 14Ø5:       knopf = get_edit_taste(x,y);
[U7Ø] 14Ø6:     }
[YN1] 14Ø7:     if (kbhit())
[P7Ø] 14Ø8:     {
[BR4] 14Ø9:       key = toupper(get_taste());
[B52] 141Ø:       if (!zeichnen)
[P7Ø] 1411:       {
[5X1] 1412:         switch(key)
[P7Ø] 1413:         {
[7Ø4] 1414:           case 'B': knopf = ABLEGEN;
[PØ1] 1415:                     break;
[DS3] 1416:           case 'H': knopf = HOLEN;
[PØ1] 1417:                     break;
[WE4] 1418:           case 'T': knopf = TAUSCHEN;
[PØ1] 1419:                     break;
[F64] 142Ø:           case 'S': knopf = SCHWARZ;
[PØ1] 1421:                     break;
[LX3] 1422:           case 'W': knopf = WEISS;
[PØ1] 1423:                     break;
[GU3] 1424:           case 'D': knopf = DREHEN;
[PØ1] 1425:                     break;
[MD3] 1426:           case 'N': knopf = NEU;
[PØ1] 1427:                     break;
[UW3] 1428:           case 'F': knopf = FERTIG;
[PØ1] 1429:                     break;
[Ø24] 143Ø:           case 'A': knopf = ABBRUCH;
[PØ1] 1431:                     break;
[U7Ø] 1432:         }
[U7Ø] 1433:       }
[U7Ø] 1434:     }
[MD2] 1435:     if (zeichnen!=Ø)
[P7Ø] 1436:     {
[X23] 1437:       x1 = (x-161)/x_step;
[UZ1] 1438:       y1 = (y-41)/11;
[VN5] 1439:       puffer = &z_edit.posxy[x1][y1/8];
[K73] 144Ø:       if (farbe == SCHWARZ)
[P7Ø] 1441:       {
[W65] 1442:         if (!(*puffer & (char)(ØxØØ8Ø>>(y1&
7))))
[P7Ø] 1443:         {
[TW4] 1444:           *puffer |= (char)(ØxØØ8Ø>>(y1&7))
;
[586] 1445:           set_edit_punkt(x1,y1,x_step,BLACK
);
[961] 1446:           if (x1>Ø)
[P7Ø] 1447:           {
[855] 1448:             *(puffer-3) &= (char)(Øxff7f>>(
y1&7));
```

```
[7A6] 1449:                 set_edit_punkt(x1-1,y1,x_step,W
HITE);
[U7Ø] 145Ø:               }
[4H2] 1451:               if (x1<spalten-1)
[P7Ø] 1452:               {
[455] 1453:                 *(puffer+3) &= (char)(Øxff7f>>(
y1&7));
[1A6] 1454:                 set_edit_punkt(x1+1,y1,x_step,W
HITE);
[U7Ø] 1455:               }
[U7Ø] 1456:             }
[U7Ø] 1457:           }
[ZRØ] 1458:           else
[P7Ø] 1459:           {
[F85] 146Ø:             if (*puffer & (char)(ØxØØ8Ø>>(y1&7)
))
[P7Ø] 1461:             {
[DV4] 1462:               *puffer &= (char)(Øxff7f>>(y1&7))
;
[8A6] 1463:               set_edit_punkt(x1,y1,x_step,WHITE
);
[U7Ø] 1464:             }
[U7Ø] 1465:           }
[U7Ø] 1466:         }
[7C2] 1467:         if (knopf!=NIX)
[P7Ø] 1468:         {
[U12] 1469:           mouse_off();
[MZ5] 147Ø:           if ((knopf!=SCHWARZ) && (knopf!=WEISS
))
[SK4] 1471:             invert_edit_taste(knopf);
[XA9] 1472:           if (((knopf==SCHWARZ) || (knopf==WEIS
S)) && (knopf!=farbe))
[P7Ø] 1473:           {
[FG4] 1474:             invert_edit_taste(farbe);
[Ø92] 1475:             farbe = knopf;
[SK4] 1476:             invert_edit_taste(knopf);
[WG6] 1477:             if (knopf==WEISS) mouse_pfeil_weiss
();
[UB3] 1478:             else mouse_pfeil();
[U7Ø] 1479:           }
[DP2] 148Ø:           if (knopf==HOLEN)
[P7Ø] 1481:           {
[VD6] 1482:             memcpy(z_edit.posxy,ablage.posxy,11
1);
[FAB] 1483:             set_raster_char(x_step,ablage_form,
spalten,(char *)z_edit.posxy);
[696] 1484:             spalten = alt_spalten = ablage_form
;
[SK4] 1485:             invert_edit_taste(knopf);
[U7Ø] 1486:           }
[B33] 1487:           if (knopf==ABLEGEN)
[P7Ø] 1488:           {
[KF6] 1489:             memcpy(ablage.posxy,z_edit.posxy,11
1);
[YV3] 149Ø:             ablage_form = spalten;
[6V8] 1491:             print_zeichen(42,98,(char *)ablage.
posxy,ablage_form);
[SK4] 1492:             invert_edit_taste(knopf);
[U7Ø] 1493:           }
[LB3] 1494:           if (knopf==TAUSCHEN)
[P7Ø] 1495:           {
[UT6] 1496:             memcpy(z_ablage.posxy,z_edit.posxy,
111);
[VD6] 1497:             memcpy(z_edit.posxy,ablage.posxy,11
1);
[YS6] 1498:             memcpy(ablage.posxy,z_ablage.posxy,
111);
[FG4] 1499:             alt_spalten = ablage_form;
[YV3] 15ØØ:             ablage_form = spalten;
[8W3] 15Ø1:             spalten = alt_spalten;
[FAB] 15Ø2:             set_raster_char(x_step,spalten,abla
ge_form,(char *)z_edit.posxy);
[6V8] 15Ø3:             print_zeichen(42,98,(char *)ablage.
posxy,ablage_form);
[SK4] 15Ø4:             invert_edit_taste(knopf);
[U7Ø] 15Ø5:           }
[AE2] 15Ø6:           if (knopf==NEU)
[P7Ø] 15Ø7:           {
[RF4] 15Ø8:             memset(z_edit.posxy,Ø,111);
[GLA] 15Ø9:             set_raster_char(x_step,spalten,spal
ten,(char *)z_edit.posxy);
[SK4] 151Ø:             invert_edit_taste(knopf);
[U7Ø] 1511:           }
[EW2] 1512:           if (knopf==SPALTE)
[P7Ø] 1513:           {
[W74] 1514:             spalten = (x-167)/x_step+1;
[HU4] 1515:             format.d1[wahl-32] = spalten;
[EØ6] 1516:             if (format.d1[wahl-32]!=alt_spalten
)
[MN9] 1517:               set_raster_char(x_step,format.d1[
wahl-32],alt_spalten,
[1L3] 1518:                               (char *)z_edit.po
sxy);
[8W3] 1519:             alt_spalten = spalten;
[BM4] 152Ø:             spalten = format.d1[wahl-32];
[U7Ø] 1521:           }
[363] 1522:           if (knopf == RECHTS)
[P7Ø] 1523:           {
```

```
[K58] 1524:             memmove(&z_edit.posxy[1][Ø],z_edit.
posxy,1Ø8);
[4FB] 1525:             z_edit.posxy[Ø][Ø] = z_edit.posxy[Ø
][1] = z_edit.posxy[Ø][2] = Ø;
[GLA] 1526:             set_raster_char(x_step,spalten,spal
ten,(char *)z_edit.posxy);
[SK4] 1527:             invert_edit_taste(knopf);
[U7Ø] 1528:           }
[YZ2] 1529:           if (knopf == LINKS)
[P7Ø] 153Ø:           {
[N18] 1531:             memmove(z_edit.posxy,&z_edit.posxy[
1][Ø],1Ø8);
[515] 1532:             z_edit.posxy[spalten-1][Ø] =
[715] 1533:               z_edit.posxy[spalten-1][1] =
[5F5] 1534:               z_edit.posxy[spalten-1][2] = Ø;
[GLA] 1535:             set_raster_char(x_step,spalten,spal
ten,(char *)z_edit.posxy);
[SK4] 1536:             invert_edit_taste(knopf);
[U7Ø] 1537:           }
[RP2] 1538:           if (knopf == HOCH)
[P7Ø] 1539:           {
[KU2] 154Ø:             for(i=Ø ; i<37 ; i++)
[P7Ø] 1541:             {
[S74] 1542:               z_edit.posxy[i][Ø] <<= 1;
[269] 1543:               if (z_edit.posxy[i][1] & 128) z_e
dit.posxy[i][Ø] |= 1;
[U74] 1544:               z_edit.posxy[i][1] <<= 1;
[869] 1545:               if (z_edit.posxy[i][2] & 128) z_e
dit.posxy[i][1] |= 1;
[M64] 1546:               z_edit.posxy[i][2] <<=1;
[U7Ø] 1547:             }
[GLA] 1548:             set_raster_char(x_step,spalten,spal
ten,(char *)z_edit.posxy);
[SK4] 1549:             invert_edit_taste(knopf);
[U7Ø] 155Ø:           }
[N73] 1551:           if (knopf == RUNTER)
[P7Ø] 1552:           {
[KU2] 1553:             for(i=Ø ; i<37 ; i++)
[P7Ø] 1554:             {
[284] 1555:               z_edit.posxy[i][2] >>= 1;
[8Ø9] 1556:               if (z_edit.posxy[i][1] & 1) z_edi
t.posxy[i][2] |= 128;
[Ø84] 1557:               z_edit.posxy[i][1] >>= 1;
[4Ø9] 1558:               if (z_edit.posxy[i][Ø] & 1) z_edi
t.posxy[i][1] |= 128;
[P64] 1559:               z_edit.posxy[i][Ø] >>=1;
[U7Ø] 156Ø:             }
[GLA] 1561:             set_raster_char(x_step,spalten,spal
ten,(char *)z_edit.posxy);
[SK4] 1562:             invert_edit_taste(knopf);
[U7Ø] 1563:           }
[E43] 1564:           if (knopf == DREHEN)
[P7Ø] 1565:           {
[MT4] 1566:             memset(z_puffer.posxy,Ø,111);
[285] 1567:             for(y=Ø ; y<24 ; y++)         /* dre
hen */
[EY2] 1568:               for(x=Ø ; x<24 ; x++)
[RRC] 1569:                 z_puffer.posxy[y][2-x/8] |= (z_
edit.posxy[x][y/8] & (128>>(y&7)))? 1<<(x&7
) : Ø;
[ØEA] 157Ø:             for(y=Ø ; y<22 ; y++)         /* unte
reinander liegende Punkte setzen */
[EY2] 1571:               for(x=Ø ; x<24 ; x++)
[X6G] 1572:                 if ((z_puffer.posxy[x][y/8] & (
128>>(y&7))) && (z_puffer.posxy[x][((y&7)<6
)? y/8 : y/8+1] & (128>>((y-6)&7))))
[P7Ø] 1573:                 {
[1R1] 1574:                   if ((y&7) < 7)
[ZP6] 1575:                     z_puffer.posxy[x][y/8] |= 1
28>>((y&7)+1);
[ZRØ] 1576:                   else
[X27] 1577:                     z_puffer.posxy[x][y/8+1] |=
 128>>((y-7)&7);
[U7Ø] 1578:                 }
[BTA] 1579:             for(y=Ø ; y<24 ; y++)         /* nebe
neinander liegende Punkte löschen */
[EY2] 158Ø:               for(x=Ø ; x<24 ; x++)
[DRD] 1581:                 if ((z_puffer.posxy[x][y/8] & (
128>>(y&7))) && (z_puffer.posxy[x+1][y/8] &
 (128>>(y&7))))
[WF6] 1582:                   z_puffer.posxy[x+1][y/8] -= 1
28>>(y&7);
[ZY6] 1583:             memcpy(z_edit.posxy,z_puffer.posxy,
111);
[GLA] 1584:             set_raster_char(x_step,spalten,spal
ten,(char *)z_edit.posxy);
[SK4] 1585:             invert_edit_taste(knopf);
[U7Ø] 1586:           }
[PØ2] 1587:           mouse_on();
[U7Ø] 1588:         }
[VY6] 1589:       } while((knopf!=FERTIG) && (knopf!=ABBRUC
H));
[U12] 159Ø:       mouse_off();
[FD5] 1591:       putimage(1Ø,1Ø,picptr1,COPY_PUT);
[YM5] 1592:       putimage(1Ø,176,picptr2,COPY_PUT);
```

Listing 1. »ze.c« enthält das Hauptprogramm des Zeichensatzeditors

```
[8V2] 1593:    farfree(picptr2);
[4V2] 1594:    farfree(picptr1);
[N33] 1595:    if (knopf == FERTIG)
[P7Ø] 1596:    {
[H88] 1597:      memcpy(zeichen[wahl-32].posxy,z_edit.po
          sxy,111);
[P23] 1598:      x = ((wahl-32)&7)*5Ø+13;
[C53] 1599:      y =  ((wahl-32)>>3)*27+4;
[D1A] 16ØØ:      print_zeichen(x,y,(char *)zeichen[wahl-
          32].posxy,spalten);
[Ø97] 16Ø1:    } else format.d1[wahl-32] = spalten_puffe
          r;
[VG2] 16Ø2:    mouse_pfeil();
[PØ2] 16Ø3:    mouse_on();
[U7Ø] 16Ø4: }
[ØØØ] 16Ø5:
[4U6] 16Ø6: /*.....................................
          ...............*/
[ØØØ] 16Ø7:
[1CA] 16Ø8: void set_raster_char(int x_step,       /* Br
          eite eines Kastens */
[7G6] 16Ø9:                      int spalten,      /* An
          zahl der Spalten */
[4S8] 161Ø:                      int alt_spalten,  /* An
          zahl der vorherigen Spalten */
[DC6] 1611:                      char *z_edit)      /* D
          aten des Zeichens */
[ØØØ] 1612:
[1E6] 1613: /* Zeichenraster und Zeichen ausgeben */
[P7Ø] 1614: {
[TA8] 1615:   int x_abbr,     /* höchste X-Position vom
          Zeichen */
[S79] 1616:       x_alt,     /* höchste vorherige X-Posi
          tion vom Zeichen */
[3E3] 1617:       x,y;       /* Bildpunkt */
[ØØØ] 1618:
[4S4] 1619:   x_abbr = 161+x_step*spalten;
[6D5] 162Ø:   x_alt = 161+x_step*alt_spalten;
[Y65] 1621:   if (schriftart == PROPORTIONAL)
[P7Ø] 1622:   {
[CT2] 1623:     setcolor(WHITE);
[GD4] 1624:     outtextxy(x_alt-4,325,"^");
[U7Ø] 1625:   }
[5R2] 1626:   setcolor(BLACK);
[865] 1627:   for(x=16Ø ; x<x_abbr ; x+=x_step)
[1G2] 1628:     line(x,4Ø,x,3Ø4);
[ØA3] 1629:   for(y=4Ø ; y<3Ø5 ; y+=11)
[5K3] 163Ø:     line(16Ø,y,x_abbr-1,y);
[K54] 1631:   if (alt_spalten>spalten)
[P7Ø] 1632:   {
[EV5] 1633:     setfillstyle(SOLID_FILL,YELLOW);
[KØ4] 1634:     bar(x_abbr,4Ø,x_alt,3Ø4);
[U7Ø] 1635:   }
[AL3] 1636:   line(x_abbr,73,626,73);
[4U3] 1637:   line(x_abbr,128,626,128);
[KT3] 1638:   line(x_abbr,216,626,216);
[ØT3] 1639:   line(x_abbr,26Ø,626,26Ø);
[Y65] 164Ø:   if (schriftart == PROPORTIONAL)
[ND4] 1641:     outtextxy(x_abbr-4,325,"^");
[416] 1642:   print_zeichen(42,4Ø,z_edit,spalten);
[T37] 1643:   print_edit_char(x_step,spalten,z_edit);
[U7Ø] 1644: }
[ØØØ] 1645:
[4U6] 1646: /*.....................................
          ...............*/
[ØØØ] 1647:
[UCA] 1648: void print_edit_char(int x_step,       /* Br
          eite eines Pixels */
[7G6] 1649:                      int spalten,      /* An
          zahl der Spalten */
[DC6] 165Ø:                      char *z_edit)      /* D
          aten des Zeichens */
[ØØØ] 1651:
[SD6] 1652: /* Gibt das Zeichen zum editieren aus */
[P7Ø] 1653: {
[BV4] 1654:   int x,y,          /* Bildposition */
[336] 1655:       ypos,         /* Y-Position des Zeiche
          ns */
[1K5] 1656:       bit;          /* einzelnes Bit prüfen *
          /
[WK7] 1657:   char puffer;     /* einzelnes Byte des Zei
          chens */
[ØØØ] 1658:
[944] 1659:   for(x=Ø ; x<spalten ; x++)
[P7Ø] 166Ø:   {
[EA1] 1661:     ypos = Ø;
[2W2] 1662:     for(y=Ø ; y<3 ; y++)
[P7Ø] 1663:     {
[H14] 1664:       puffer = z_edit[x*3+y];
[2L4] 1665:       for(bit=128 ; bit>Ø ; bit >>= 1)
[P7Ø] 1666:       {
[NC2] 1667:         if (puffer&bit)
[2C6] 1668:           set_edit_punkt(x,ypos,x_step,BLAC
          K);
[ZRØ] 1669:         else
[9E6] 167Ø:           set_edit_punkt(x,ypos,x_step,WHIT
          E);
[M71] 1671:         ypos++;
[U7Ø] 1672:       }
[U7Ø] 1673:     }
[U7Ø] 1674:   }
[U7Ø] 1675: }
[ØØØ] 1676:
[4U6] 1677: /*.....................................
          ...............*/
[ØØØ] 1678:
[F2A] 1679: void set_edit_punkt(int x,int y,     /* Posi
          tion des Pixels */
[MX5] 168Ø:                      int x_step,       /* Bre
          ite des Pixels */
[BF5] 1681:                      int farbe)        /* Farb
          e des Pixels */
[ØØØ] 1682:
[7BC] 1683: /* Gibt einen Punkt des zu editierenden Zei
          chens auf den Bildschirm aus */
[P7Ø] 1684: {
[U12] 1685:   mouse_off();
[TN5] 1686:   setfillstyle(SOLID_FILL,farbe);
[ØB8] 1687:   bar(x*x_step+161,y*11+41,x*x_step+159+x_s
          tep,y*11+5Ø);
[PØ2] 1688:   mouse_on();
[U7Ø] 1689: }
[ØØØ] 1690:
[4U6] 1691: /*.....................................
          ...............*/
[ØØØ] 1692:
[1DA] 1693: void invert_edit_taste(int wahl)     /* zu i
          nvertierender Knopf */
[ØØØ] 1694:
[8Z5] 1695: /* Invertiert den angegebenen Knopf */
[P7Ø] 1696: {
[B12] 1697:   switch(wahl)
[P7Ø] 1698:   {
[P96] 1699:     case ABLEGEN:  invertimage(2Ø,125,1Ø6,1
          39);
[PØ1] 17ØØ:                    break;
[U76] 17Ø1:     case HOLEN:    invertimage(2Ø,141,1Ø6,1
          55);
[PØ1] 17Ø2:                    break;
[6D6] 17Ø3:     case TAUSCHEN:invertimage(2Ø,157,1Ø6,17
          1);
[PØ1] 17Ø4:                    break;
[2E6] 17Ø5:     case SCHWARZ:  invertimage(2Ø,183,1Ø6,1
          97);
[PØ1] 17Ø6:                    break;
[S96] 17Ø7:     case WEISS:    invertimage(2Ø,199,1Ø6,2
          13);
[PØ1] 17Ø8:                    break;
[D76] 17Ø9:     case DREHEN:   invertimage(2Ø,225,1Ø6,23
          9);
[PØ1] 171Ø:                    break;
[FZ5] 1711:     case NEU:      invertimage(2Ø,251,1Ø6,2
          65);
[PØ1] 1712:                    break;
[U76] 1713:     case FERTIG:   invertimage(2Ø,277,1Ø6,29
          1);
[PØ1] 1714:                    break;

[AB6] 1715:     case ABBRUCH:  invertimage(2Ø,293,1Ø6,3
          Ø7);
[PØ1] 1716:                    break;
[W56] 1717:     case HOCH:     invertimage(126,183,142,1
          97);
[PØ1] 1718:                    break;
[TG6] 1719:     case RUNTER:   invertimage(126,199,142,2
          13);
[PØ1] 172Ø:                    break;
[Z86] 1721:     case LINKS:    invertimage(111,191,127,
          2Ø5);
[PØ1] 1722:                    break;
[XE6] 1723:     case RECHTS:   invertimage(141,191,157,2
          Ø5);
[PØ1] 1724:                    break;
[U7Ø] 1725:   }
[U7Ø] 1726: }
[ØØØ] 1727:
[4U6] 1728: /*.....................................
          ...............*/
[ØØØ] 1729:
[4M9] 173Ø: unsigned get_edit_taste(int x,int y)    /*
          Mausposition */
[ØØØ] 1731:
[4KB] 1732: /* Liefert aus der gegebenen Mausposition d
          ie Nr. der gewählten Taste */
[P7Ø] 1733: {
[2M2] 1734:   if ((x>21) && (x<1Ø5))
[P7Ø] 1735:   {
[9L5] 1736:     if ((y>126) && (y<138)) return(ABLEGEN)
          ;
[L95] 1737:     if ((y>142) && (y<154)) return(HOLEN);
[CX5] 1738:     if ((y>158) && (y<17Ø)) return(TAUSCHEN
          );
[TR5] 1739:     if ((y>184) && (y<196)) return(SCHWARZ)
          ;
[HA5] 174Ø:     if ((y>2ØØ) && (y<212)) return(WEISS);
[PH5] 1741:     if ((y>226) && (y<238)) return(DREHEN);
[2Y4] 1742:     if ((y>252) && (y<264)) return(NEU);
```

```
[HL5] 1743:      if ((y>278) && (y<29Ø)) return(FERTIG);
[1M5] 1744:      if ((y>294) && (y<3Ø6)) return(ABBRUCH)
          ;
[U7Ø] 1745:    }
[XW2] 1746:    if ((x>127) && (x<141))
[P7Ø] 1747:    {
[845] 1748:      if ((y>184) && (y<196)) return(HOCH);
[TL5] 1749:      if ((y>2ØØ) && (y<212)) return(RUNTER);
[U7Ø] 1750:    }
[7X2] 1751:    if ((y>192) && (y<2Ø4))
[P7Ø] 1752:    {
[4A5] 1753:      if ((x>112) && (x<126)) return(LINKS);
[9H5] 1754:      if ((x>142) && (x<156)) return(RECHTS);
[U7Ø] 1755:    }
[3GA] 1756:    if ((schriftart==PROPORTIONAL) && (x>167)
          && (x<574) && (y>323) && (y<333))
[HL2] 1757:      return(SPALTE);
[DØ2] 1758:    return(NIX);
[U7Ø] 1759: }
[ØØØ] 176Ø:
[4U6] 1761: /*.....................................
          ...............*/
[ØØØ] 1762:
[H54] 1763: int initialisieren(void)
[ØØØ] 1764:
[VB5] 1765: /* Initialisieren: - Graphikkarte
[A62] 1766:                    - Maustreiber
[862] 1767:                    - Zeichensatz
[RX1] 1768:                    - Variablen
[AU4] 1769:                    - Fehlerbehandlungsrouti
          ne */
[P7Ø] 177Ø: {
[2M6] 1771:   int graphdriver,       /* Graphiktreiber *
          /
[Y55] 1772:       graphmode;        /* Graphikmodus */
[KX7] 1773:   union REGS regs;       /* Variablen für In
          terrupt */
[ØØØ] 1774:
[NUA] 1775:   start_pfad[Ø]='A'+getdisk();
          /* aktuelles Verzeichnis nach */
[Ø39] 1776:   start_pfad[1]=':';
          /* dem Start festhalten */
[HD3] 1777:   start_pfad[2]='\\';
[9U4] 1778:   getcurdir(Ø,&start_pfad[3]);
[KW5] 1779:   registerbgidriver(EGAVGA_driver);
[KM5] 178Ø:   registerbgidriver(Herc_driver);
[FB5] 1781:   registerbgidriver(small_font);
[RF2] 1782:   regs.h.ah = Øx0f;
[7L3] 1783:   int86(Øx1Ø,&regs,&regs);
[7F2] 1784:   if (regs.h.al==7)
[P7Ø] 1785:   {
[C24] 1786:     graphdriver = HERCMONO;
[WØ4] 1787:     graphmode = HERCMONOHI;
[F11] 1788:   } else
[P7Ø] 1789:   {
[GØ3] 179Ø:     graphdriver = EGA;
[VX2] 1791:     graphmode = EGAHI;
[U7Ø] 1792:   }
[U36] 1793:   initgraph(&graphdriver,&graphmode,"");
[RØ9] 1794:   if ((graphdriver<Ø) || (getmaxx()<639) ||
          (getmaxy()<347))
[P7Ø] 1795:   {
[TH5] 1796:     if (graphdriver >= Ø) closegraph();
[PD2] 1797:     return(ERROR);
[U7Ø] 1798:   }
[CC6] 1799:   if (init_graph_mouse(graphdriver)==Ø)
[P7Ø] 18ØØ:   {
[S42] 18Ø1:     closegraph();
[MNE] 18Ø2:     printf("\nMaustreiber nicht vorhanden.\
          nBitte zuerst Maustreiber installieren.\n")
          ;
[PD2] 18Ø3:     return(ERROR);
[U7Ø] 18Ø4:   }
[UFD] 18Ø5:   settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
          /* Graphikzeichensatz installieren */
[YP3] 18Ø6:   CLS_ZEICHEN(zeichen);
[PW1] 18Ø7:   datei[Ø]=Ø;
[W93] 18Ø8:   schriftart = DRAFT;
[AP2] 18Ø9:   reset_format();
[174] 181Ø:   drucker_mode = KOMPLETT;
[D1A] 1811:   harderr(handler);          /* Fehlermeldu
          ng von MS/DOS abfangen */
[Z57] 1812:   setfillstyle(INTERLEAVE_FILL,LIGHTGRAY);
[T82] 1813:   bar(Ø,Ø,639,347);
[KW1] 1814:   return(OK);
[U7Ø] 1815: }
[ØØØ] 1816:
[4U6] 1817: /*.....................................
          ...............*/
[ØØØ] 1818:
[S54] 1819: void reset_format(void)
[ØØØ] 182Ø:
[MR6] 1821: /* Zeichensatzformat Grundeinstellung */
[P7Ø] 1822: {
[D32] 1823:   format.dØ = Ø;
[433] 1824:   switch(schriftart)
[P7Ø] 1825:   {
[U84] 1826:     case DRAFT:  format.d2 = 3;
[RM3] 1827:                  memset(format.d1,9,96);
[PØ1] 1828:                  break;
[884] 1829:     case ELITE:  format.d2 = 7;
[HN3] 183Ø:                  memset(format.d1,23,96);
[PØ1] 1831:                  break;
[ØY3] 1832:     case PICA:   format.d2 = 7;
[9P3] 1833:                  memset(format.d1,29,96);
[PØ1] 1834:                  break;
[3H5] 1835:     case PROPORTIONAL:  format.d2 = 5;
[4P3] 1836:                  memset(format.d1,37,96);
[PØ1] 1837:                  break;
[U7Ø] 1838:   }
[U7Ø] 1839: }
[ØØØ] 184Ø:
[4U6] 1841: /*.....................................
          ...............*/
[ØØØ] 1842:
[CY3] 1843: void statuszeile(void)
[ØØØ] 1844:
[844] 1845: /* Statuszeile ausgeben */
[P7Ø] 1846: {
[VZ6] 1847:   char name[MAXFILE+MAXEXT],     /* Dateinam
          e */
[F46] 1848:        ext[MAXEXT],              /* Dateisuf
          fix */
[P76] 1849:        text[4];                  /* Stringsp
          eicher */
[ØØØ] 185Ø:
[AD6] 1851:   setfillstyle(SOLID_FILL,LIGHTBLUE);
[YK2] 1852:   bar(Ø,335,639,347);
[367] 1853:   if (getgraphmode() == EGAHI) setcolor(WHI
          TE);
[SG3] 1854:                              else setcolor(BLA
          CK);
[P44] 1855:   outtextxy(8,338,"Datei:");
[UW5] 1856:   fnsplit(datei,NULL,NULL,name,ext);
[4Y5] 1857:   outtextxy(62,338,strcat(name,ext));
[EZ4] 1858:   sprintf(text,"%Ø2i",format.dØ);
[WY6] 1859:   outtextxy(3ØØ,338,"Zeichenabstand: links
          =");
[9X3] 186Ø:   outtextxy(492,338,text);
[KZ4] 1861:   sprintf(text,"%Ø2i",format.d2);
[8W4] 1862:   outtextxy(516,338,"/ rechts =");
[CX3] 1863:   outtextxy(6Ø4,338,text);
[U7Ø] 1864: }
[ØØØ] 1865:
[4U6] 1866: /*.....................................
          ...............*/
[ØØØ] 1867:
[NP3] 1868: void titelzeile(void)
[ØØØ] 1869:
[WG4] 187Ø: /* Titel und Menü ausgeben */
[P7Ø] 1871: {
[WK9] 1872:   putbox(419,Ø,639,113,"Zeichensatz Editor"
          ,"",BLACK,LIGHTBLUE);
[5R2] 1873:   setcolor(BLACK);
[YS2] 1874:   line(517,15,517,11Ø);
[U35] 1875:   set_knopf(424,17,514,29,"(L)aden");
[BX5] 1876:   set_knopf(424,33,514,45,"(S)peichern");
[UC5] 1877:   set_knopf(424,49,514,61,"(D)rucker");
[H95] 1878:   set_knopf(424,65,514,77,"(I)mport");
[AR4] 1879:   set_knopf(424,81,514,93,"(N)eu");
[FZ4] 188Ø:   set_knopf(424,97,514,1Ø9,"(E)nde");
[W15] 1881:   set_knopf(52Ø,17,625,29,"D(r)aft");
[M35] 1882:   set_knopf(52Ø,33,625,45,"Eli(t)e");
[HX4] 1883:   set_knopf(52Ø,49,625,61,"(P)ica");
[716] 1884:   set_knopf(52Ø,65,625,77,"Pr(o)portio.");
[6G6] 1885:   set_knopf(52Ø,89,634,1Ø1,"(Z)wischenraum"
          );
[5R2] 1886:   setcolor(BLACK);
[RM2] 1887:   line(626,23,631,23);
[6P2] 1888:   line(626,39,631,39);
[MN2] 1889:   line(626,55,631,55);
[2N2] 189Ø:   line(626,71,631,71);
[ZM2] 1891:   line(632,23,632,71);
[T44] 1892:   invertwahl(schriftart);
[2G3] 1893:   putbox(419,13Ø,639,28Ø,"",
[VG8] 1894:         "Zum Editieren klicken Sie|bitte da
          s Zeichen an...",
[7W2] 1895:         BLACK,LIGHTCYAN);
[ZB6] 1896:   outtextxy(453,216,"Zeichensatz Editor");
[RH4] 1897:   outtextxy(465,228,VERSION);
[9Ø6] 1898:   outtextxy(453,24Ø,"(c) by Markus Graf");
[FZ4] 1899:   outtextxy(453,252,"    March 199Ø");
[ET3] 19ØØ:   rectangle(44Ø,212,61Ø,265);
[U7Ø] 1901: }
[ØØØ] 1902:
[4U6] 1903: /*.....................................
          ...............*/
[ØØØ] 1904:
[SZ8] 1905: int get_menue_taste(int x,int y)    /* Maus
```

Listing 1. »ze.c« enthält das Hauptprogramm des Zeichensatzeditors

```
            position */
[ØØØ] 19Ø6:
[519] 19Ø7: /* liefert einen evtl. gewählten Menüpunkt
            zurück */
[P7Ø] 19Ø8: {
[6X2] 19Ø9:   if ((x>52Ø) && (x<625))
[P7Ø] 191Ø:   {
[735] 1911:     if ((y>17) && (y<29)) return(DRAFT);
[T15] 1912:     if ((y>33) && (y<45)) return(ELITE);
[YY4] 1913:     if ((y>49) && (y<61)) return(PICA);
[UK6] 1914:     if ((y>65) && (y<77)) return(PROPORTION
AL);
[U7Ø] 1915:   }
[W59] 1916:   if ((x>52Ø) && (x<634) && (y>89) && (y<1Ø
1)) return(ZWISCHENRAUM);
[AX2] 1917:   if ((x>424) && (x<514))
[P7Ø] 1918:   {
[225] 1919:     if ((y>17) && (y<29))  return(LADEN);
[ZT5] 192Ø:     if ((y>33) && (y<45))  return(SPEICHERN
);
[7F5] 1921:     if ((y>49) && (y<61))  return(DRUCKER);
[8A5] 1922:     if ((y>65) && (y<77))  return(IMPORT);
[9T4] 1923:     if ((y>81) && (y<93))  return(NEU);
[7U4] 1924:     if ((y>97) && (y<1Ø9)) return(ENDE);
[U7Ø] 1925:   }
[T15] 1926:   if ((x>2) && (x<4Ø1) && (y>2) && (y<325))
[3A4] 1927:     return((x-2)/5Ø+(y-2)/27*8+32);
[DØ2] 1928:   return(NIX);
[U7Ø] 1929: }
[ØØØ] 193Ø:
[4U6] 1931: /*.....................................
            ...............*/
[ØØØ] 1932:
[858] 1933: void invertwahl(int wahl)    /* gewählter K
            nopf */
[ØØØ] 1934:
[677] 1935: /* gewählter Menüpunkt wird invertiert */
[P7Ø] 1936: {
[PP4] 1937:   int x,y;       /* Bildposition */
[ØØØ] 1938:
[U12] 1939:   mouse_off();
[B12] 194Ø:   switch(wahl)
[P7Ø] 1941:   {
[N16] 1942:     case DRAFT:  invertimage(519,16,626,3Ø)
;
[PØ1] 1943:                  break;
[116] 1944:     case ELITE:  invertimage(519,32,626,46)
;
[PØ1] 1945:                  break;
[SX5] 1946:     case PICA:   invertimage(519,48,626,62);
[PØ1] 1947:                  break;
[4H7] 1948:     case PROPORTIONAL:  invertimage(519,64,
626,78);
[PØ1] 1949:                  break;
[RF7] 195Ø:     case ZWISCHENRAUM:  invertimage(519,88,
635,1Ø2);
[PØ1] 1951:                  break;

[AZ5] 1952:     case LADEN:  invertimage(423,16,515,3Ø)
;
[PØ1] 1953:                  break;
[UT6] 1954:     case SPEICHERN: invertimage(423,32,515,
46);
[PØ1] 1955:                  break;
[F76] 1956:     case DRUCKER:  invertimage(423,48,515,6
2);
[PØ1] 1957:                  break;
[A76] 1958:     case IMPORT:  invertimage(423,64,515,78
);
[PØ1] 1959:                  break;
[YS5] 196Ø:     case NEU:    invertimage(423,8Ø,515,94)
;
[PØ1] 1961:                  break;
[WW5] 1962:     case ENDE:  invertimage(423,96,515,11Ø)
;
[PØ1] 1963:                  break;
[U7Ø] 1964:   }
[FT3] 1965:   if ((wahl>31) && (wahl<128))
[P7Ø] 1966:   {
[123] 1967:     x = ((wahl-32)&7)*5Ø+3;
[333] 1968:     y = ((wahl-32)/8)*27+3;
[FE4] 1969:     invertimage(x,y,x+49,y+26);
[U7Ø] 197Ø:   }
[PØ2] 1971:   mouse_on();
[U7Ø] 1972: }
[ØØØ] 1973:
[4U6] 1974: /*.....................................
            ...............*/
[ØØØ] 1975:
[L1C] 1976: void zeichensatz_tabelle(z_bytes *zeichensa
            tz,    /* Daten Zeichensatz */
[GK6] 1977:                              form format)
            /* Breite der Zeichen */
[ØØØ] 1978:
[LT8] 1979: /* gibt die Zeichensatztabelle auf den Bild
            schirm aus */
```

```
[P7Ø] 198Ø: {
[B55] 1981:   int x,y,            /* Bildposition */
[HA4] 1982:       n,              /* Zeichennummer */
[1Ø5] 1983:       spalten;        /* Zeichenbreite */
[SH7] 1984:   char ascii[2];      /* String Zwischenspe
            icher */
[ØØØ] 1985:
[433] 1986:   switch(schriftart)
[P7Ø] 1987:   {
[NY3] 1988:     case DRAFT:  spalten = 9;
[PØ1] 1989:                  break;
[1Z3] 199Ø:     case ELITE: spalten = 23;
[PØ1] 1991:                  break;
[7U3] 1992:     case PICA:  spalten = 29;
[PØ1] 1993:                  break;
[U7Ø] 1994:   }
[U12] 1995:   mouse_off();
[VM5] 1996:   setfillstyle(SOLID_FILL,BLACK);
[P72] 1997:   bar(Ø,Ø,4Ø4,328);
[1R3] 1998:   setcolor(LIGHTGREEN);
[U93] 1999:   rectangle(Ø,Ø,4Ø4,328);
[W93] 2ØØØ:   rectangle(1,1,4Ø3,327);
[Y93] 2ØØ1:   rectangle(2,2,4Ø2,326);
[DL3] 2ØØ2:   for(y=29 ; y<=299 ; y+=27)
[382] 2ØØ3:     line(Ø,y,4Ø4,y);
[MK3] 2ØØ4:   for(x=52 ; x<=352 ; x+=5Ø)
[PA2] 2ØØ5:     line(x,Ø,x,328);
[7PØ] 2ØØ6:   n=32;
[EH3] 2ØØ7:   for(y=4 ; y<328 ; y+=27)
[7A3] 2ØØ8:     for(x=13 ; x<413 ; x+=5Ø)
[P7Ø] 2ØØ9:     {
[YH9] 2Ø1Ø:       if (schriftart==PROPORTIONAL) spalten
 = format.dl[n-32];
[ØDA] 2Ø11:       print_zeichen(x,y,(char *)zeichensatz
            [n-32].posxy,spalten);
[KØ3] 2Ø12:       bar(x-9,y,x-1,y+23);
[5R2] 2Ø13:       setcolor(BLACK);
[ØZ1] 2Ø14:       ascii[Ø]=n;
[AW1] 2Ø15:       ascii[1]=Ø;
[144] 2Ø16:       outtextxy(x-9,y,ascii);
[4GØ] 2Ø17:       n++;
[U7Ø] 2Ø18:     }
[PØ2] 2Ø19:   mouse_on();
[U7Ø] 2Ø2Ø: }
[ØØØ] 2Ø21:
[4U6] 2Ø22: /*.....................................
            ...............*/
[ØØØ] 2Ø23:
[119] 2Ø24: void print_zeichen(int x,int y,          /*
             Bildposition */
[XW6] 2Ø25:                    char *zeichensatz,    /*
            Daten des Zeichens */
[PØ7] 2Ø26:                    int spalten)          /*
            Breite des Zeichens */
[ØØØ] 2Ø27:
[CY9] 2Ø28: /* gibt ASCII-Zeichen n auf die Bildschirma
            dresse (x,y) aus */
[P7Ø] 2Ø29: {
[YF8] 2Ø3Ø:   int xpos,ypos,          /* Byteposition d
            es Zeichens */
[AR5] 2Ø31:       xpixel,ypixel,      /* Bildposition *
            /
[926] 2Ø32:       bit;                /* Testet, ob Bit
             gesetzt */
[HG8] 2Ø33:    unsigned char puffer;    /* Zeiger auf By
            teposition */
[ØØØ] 2Ø34:
[BE3] 2Ø35:   xpixel=x; ypixel=y;
[YP5] 2Ø36:   setfillstyle(SOLID_FILL,WHITE);
[WX2] 2Ø37:   bar(x,y,x+36,y+23);
[5R2] 2Ø38:   setcolor(BLACK);
[RU5] 2Ø39:   for(xpos=Ø ; xpos<spalten ; xpos++)
[P7Ø] 2Ø4Ø:   {
[9K4] 2Ø41:     for(ypos=Ø ; ypos<3 ; ypos++)
[P7Ø] 2Ø42:     {
[XV5] 2Ø43:       puffer = *(zeichensatz+3*xpos+ypos);
[HE4] 2Ø44:       for(bit=128 ; bit>Ø ; bit>>=1)
[P7Ø] 2Ø45:       {
[NC2] 2Ø46:         if (puffer&bit)
[P7Ø] 2Ø47:         {
[M7A] 2Ø48:           if (schriftart == DRAFT) line(xpi
            xel,ypixel,xpixel+3,ypixel);
[496] 2Ø49:           else putpixel(xpixel,ypixel,BLACK
            );
[U7Ø] 2Ø5Ø:         }
[TE1] 2Ø51:         ypixel++;
[U7Ø] 2Ø52:       }
[U7Ø] 2Ø53:     }
[FK1] 2Ø54:     ypixel=y;
[3M5] 2Ø55:     xpixel+=(schriftart == DRAFT)? 4 : 1;
[U7Ø] 2Ø56:   }
[U7Ø] 2Ø57: }
```

Listing 1. »ze.c« enthält das Hauptprogramm des Zeichensatzeditors

```
[Y2G]    1: /*_____________ UNIT zur Druckerinitial
         isierung _____________________
[ØØØ]    2:
[A74]    3:     Autor              : Markus Graf
[4W2]    4:     Version            : 1.2
[GM4]    5:     Zuerst erstellt    : Ø6.Ø4.199Ø
[3X4]    6:     Zuletzt geändert   : 14.Ø4.199Ø
[ØØØ]    7:
[ØØØ]    8:
[V92]    9:     Beschreibung:
[ØØØ]   1Ø:
[YEB]   11:       Alle Funktionen setzen im Fehlerfall
         die Variable druck_err = ERROR.
[ØØØ]   12:
[K88]   13:       drucker_init();        :  Drucker Ini
         tialisierung mit
[T21]   14:                                ESC '@'
[8D9]   15:       rom_in_ram();          : ROM Zeichens
         atz in RAM kopieren mit
[7L1]   16:                                ESC ':' Ø Ø Ø
[XS8]   17:       draft_mode();          : Draft Zeiche
         nsatz einschalten mit
[191]   18:                                ESC 'x' Ø
[GØ8]   19:       lq_mode();             : LQ Zeichensat
         z einschalten mit
[391]   2Ø:                                ESC 'x' 1
[YZ7]   21:       elite_mode();          : Elite Zeiche
         nbreite wählen
[Ø41]   22:                                ESC 'M'
[LA7]   23:       pica_mode();           : Pica Zeichenb
         reite wählen
[941]   24:                                ESC 'P'
[FB9]   25:       hoch_mode();           : Zeichen Hochs
         tellung einschalten mit
[K81]   26:                                ESC 'S' Ø
[7C9]   27:       tief_mode();           : Zeichen Tiefs
         tellung einschalten mit
[M81]   28:                                ESC 'S' 1
[Z1B]   29:       no_hoch_tief_mode();   : normale Zeich
         enstellung einschalten mit
[N41]   3Ø:                                ESC 'T'
[2A9]   31:       kursiv_mode();         : Kursiv Zeiche
         nsatz einschalten mit
[N11]   32:                                ESC '4'
[X5A]   33:       no_kursiv_mode();      : Kursiv Zeich
         ensatz ausschalten mit
[S11]   34:                                ESC '5'
[Z6A]   35:       breitdruck_mode();     : Zeichen mit d
         oppelter Breite ausgeben
[Ø91]   36:                                ESC 'W' 1
[K6A]   37:       no_breitdruck_mode();  :  Zeichen mit
          normaler Breite ausgeben
[Y81]   38:                                ESC 'W' Ø
[DR9]   39:       hochdruck_mode();      : Zeichen mit
         doppelter Höhe ausgeben
[Ø91]   4Ø:                                ESC 'w' 1
[ØY9]   41:       no_hochdruck_mode();   :  Zeichen mit
         normaler Höhe ausgeben
[Y81]   42:                                ESC 'w' Ø
[V6A]   43:       prop_mode();           : Proportional
         Zeichensatz einschalten mit
[B81]   44:                                ESC 'p' 1
[8MA]   45:       no_prop_mode();        : Proportional
          Zeichensatz ausschalten mit
[981]   46:                                ESC 'p' Ø
[TC7]   47:       fett_mode();           : Fettdruck ein
         schalten mit
[831]   48:                                ESC 'E'
[SU7]   49:       no_fett_mode();        : Fettdruck au
         sschalten mit
[B31]   5Ø:                                ESC 'F'
[PFB]   51:       benutz_z_satz();       : Benutzer defi
         nierter Zeichensatz einschalten;
[A41]   52:                                ESC '%' 1
[ØØØ]   53:
[17Ø]   54: */
[ØØØ]   55:
[ØØØ]   56:
[7P9]   57: /********************** Funktions-Prototyp
         en *******************************/
[ØØØ]   58:
[1E5]   59: void drucker_ausgabe(int,char *);
[N44]   6Ø: void drucker_init(void);
[RX3]   61: void rom_in_ram(void);
[HW3]   62: void draft_mode(void);
[AB3]   63: void lq_mode(void);
[LV3]   64: void elite_mode(void);
[FH3]   65: void pica_mode(void);
[XH3]   66: void hoch_mode(void);
[7K3]   67: void tief_mode(void);
[V25]   68: void no_hoch_tief_mode(void);
[H64]   69: void outline_mode(void);
[D34]   7Ø: void shadow_mode(void);
[GN5]   71: void outline_shadow_mode(void);
[A66]   72: void no_outline_shadow_mode(void);
[254]   73: void kursiv_mode(void);
[XR4]   74: void no_kursiv_mode(void);
[4T4]   75: void breitdruck_mode(void);
[8C5]   76: void no_breitdruck_mode(void);
[BL4]   77: void hochdruck_mode(void);
[R25]   78: void no_hochdruck_mode(void);
[XL3]   79: void prop_mode(void);
[L74]   8Ø: void no_prop_mode(void);
[SH3]   81: void fett_mode(void);
[G64]   82: void no_fett_mode(void);
[8G4]   83: void benutz_z_satz(void);
[ØØØ]   84:
[ØØØ]   85:
[N89]   86: /********************** Unterprogramme ***
         *********************************/
[ØØØ]   87:
[ABC]   88: void drucker_ausgabe(int n,          /* Anz
         ahl der zu übertragenden Zeichen */
[XE7]   89:                       char *zeichen)  /* Arra
         y enthält die Zeichen */
[ØØØ]   9Ø:
[U9C]   91: /* Zeichen an den Drucker senden und im Feh
         lerfall Funktion sofort beenden */
[P7Ø]   92: {
[F11]   93:   int i;
[ØØØ]   94:
[ST2]   95:   for(i=Ø ; i<n ; i++)
[P7Ø]   96:   {
[FG4]   97:     fputc(zeichen[i],stdprn);
[6W4]   98:     if (druck_err == ERROR) break;
[U7Ø]   99:   }
[U7Ø]  1ØØ: }
[ØØØ]  1Ø1:
[U24]  1Ø2: void drucker_init(void)
[P7Ø]  1Ø3: {
[BS1]  1Ø4:   char z[2];
[ØØØ]  1Ø5:
[TR1]  1Ø6:   z[Ø] = ESC;
[5L1]  1Ø7:   z[1] = '@';
[8N3]  1Ø8:   drucker_ausgabe(2,z);
[U7Ø]  1Ø9: }
[ØØØ]  11Ø:
[6S3]  111: void rom_in_ram(void)
[P7Ø]  112: {
[ES1]  113:   char z[5];
[ØØØ]  114:
[TR1]  115:   z[Ø] = ESC;
[TK1]  116:   z[1] = ':';
[LH3]  117:   z[2] = z[3] = z[4] = Ø;
[EN3]  118:   drucker_ausgabe(5,z);
[U7Ø]  119: }
[ØØØ]  12Ø:
[ØR3]  121: void draft_mode(void)
[P7Ø]  122: {
[CS1]  123:   char z[3];
[ØØØ]  124:
[TR1]  125:   z[Ø] = ESC;
[NM1]  126:   z[1] = 'x';
[781]  127:   z[2] = Ø;
[AN3]  128:   drucker_ausgabe(3,z);
[U7Ø]  129: }
[ØØØ]  13Ø:
[Y33]  131: void lq_mode(void)
[P7Ø]  132: {
[CS1]  133:   char z[3];
[ØØØ]  134:
[TR1]  135:   z[Ø] = ESC;
[NM1]  136:   z[1] = 'x';
[881]  137:   z[2] = 1;
[AN3]  138:   drucker_ausgabe(3,z);
[U7Ø]  139: }
[ØØØ]  14Ø:
[2P3]  141: void elite_mode(void)
[P7Ø]  142: {
[BS1]  143:   char z[2];
[ØØØ]  144:
[TR1]  145:   z[Ø] = ESC;
[ZL1]  146:   z[1] = 'M';
[8N3]  147:   drucker_ausgabe(2,z);
[U7Ø]  148: }
[ØØØ]  149:
[TD3]  15Ø: void pica_mode(void)
[P7Ø]  151: {
[BS1]  152:   char z[2];
[ØØØ]  153:
[TR1]  154:   z[Ø] = ESC;
[5M1]  155:   z[1] = 'P';
[8N3]  156:   drucker_ausgabe(2,z);
[U7Ø]  157: }
[ØØØ]  158:
[7E3]  159: void hoch_mode(void)
[P7Ø]  16Ø: {
[CS1]  161:   char z[3];
[ØØØ]  162:
[TR1]  163:   z[Ø] = ESC;
[BM1]  164:   z[1] = 'S';
[781]  165:   z[2] = Ø;
[AN3]  166:   drucker_ausgabe(3,z);
[U7Ø]  167: }
```

Listing 2. Mit der Unit »drucker.c« wird die Druckerinitialisierung vorgenommen

```
[ØØØ]  168:
[HE3]  169: void tief_mode(void)
[P7Ø]  17Ø: {
[CS1]  171:   char z[3];
[ØØØ]  172:
[TR1]  173:   z[Ø] = ESC;
[BM1]  174:   z[1] = 'S';
[881]  175:   z[2] = 1;
[AN3]  176:   drucker_ausgabe(3,z);
[U7Ø]  177: }
[ØØØ]  178:
[5Z4]  179: void no_hoch_tief_mode(void)
[P7Ø]  18Ø: {
[BS1]  181:   char z[2];
[ØØØ]  182:
[TR1]  183:   z[Ø] = ESC;
[DM1]  184:   z[1] = 'T';
[8N3]  185:   drucker_ausgabe(2,z);
[U7Ø]  186: }
[ØØØ]  187:
[P44]  188: void outline_mode(void)
[P7Ø]  189: {
[CS1]  19Ø:   char z[3];
[ØØØ]  191:
[TR1]  192:   z[Ø] = ESC;
[7M1]  193:   z[1] = 'q';
[881]  194:   z[2] = 1;
[AN3]  195:   drucker_ausgabe(3,z);
[U7Ø]  196: }
[ØØØ]  197:
[1W3]  198: void shadow_mode(void)
[P7Ø]  199: {
[CS1]  2ØØ:   char z[3];
[ØØØ]  2Ø1:
[TR1]  2Ø2:   z[Ø] = ESC;
[7M1]  2Ø3:   z[1] = 'q';
[981]  2Ø4:   z[2] = 2;
[AN3]  2Ø5:   drucker_ausgabe(3,z);
[U7Ø]  2Ø6: }
[ØØØ]  2Ø7:
[4E5]  2Ø8: void outline_shadow_mode(void)
[P7Ø]  2Ø9: {
[CS1]  21Ø:   char z[3];
[ØØØ]  211:
[TR1]  212:   z[Ø] = ESC;
[7M1]  213:   z[1] = 'q';
[A81]  214:   z[2] = 3;
[AN3]  215:   drucker_ausgabe(3,z);
[U7Ø]  216: }
[ØØØ]  217:
[TØ6]  218: void no_outline_shadow_mode(void)
[P7Ø]  219: {
[CS1]  22Ø:   char z[3];
[ØØØ]  221:
[TR1]  222:   z[Ø] = ESC;
[7M1]  223:   z[1] = 'q';
[781]  224:   z[2] = Ø;
[AN3]  225:   drucker_ausgabe(3,z);
[U7Ø]  226: }
[ØØØ]  227:
[PX3]  228: void kursiv_mode(void)
[P7Ø]  229: {
[BS1]  23Ø:   char z[2];
[ØØØ]  231:
[TR1]  232:   z[Ø] = ESC;
[DK1]  233:   z[1] = '4';
[8N3]  234:   drucker_ausgabe(2,z);
[U7Ø]  235: }
[ØØØ]  236:
[CK4]  237: void no_kursiv_mode(void)
[P7Ø]  238: {
[BS1]  239:   char z[2];
[ØØØ]  24Ø:
[TR1]  241:   z[Ø] = ESC;
[FK1]  242:   z[1] = '5';
[8N3]  243:   drucker_ausgabe(2,z);
[U7Ø]  244: }
[ØØØ]  245:
[SH4]  246: void breitdruck_mode(void)
[P7Ø]  247: {
[CS1]  248:   char z[3];
[ØØØ]  249:
[TR1]  25Ø:   z[Ø] = ESC;
[LM1]  251:   z[1] = 'W';
[881]  252:   z[2] = 1;
[AN3]  253:   drucker_ausgabe(3,z);
[U7Ø]  254: }
[ØØØ]  255:
[R65]  256: void no_breitdruck_mode(void)
[P7Ø]  257: {
[CS1]  258:   char z[3];
[ØØØ]  259:
[TR1]  26Ø:   z[Ø] = ESC;
[LM1]  261:   z[1] = 'W';
[781]  262:   z[2] = Ø;
[AN3]  263:   drucker_ausgabe(3,z);
[U7Ø]  264: }
[ØØØ]  265:
```

```
[UD4]  266: void hochdruck_mode(void)
[P7Ø]  267: {
[CS1]  268:   char z[3];
[ØØØ]  269:
[TR1]  27Ø:   z[Ø] = ESC;
[LM1]  271:   z[1] = 'w';
[881]  272:   z[2] = 1;
[AN3]  273:   drucker_ausgabe(3,z);
[U7Ø]  274: }
[ØØØ]  275:
[1Z4]  276: void no_hochdruck_mode(void)
[P7Ø]  277: {
[CS1]  278:   char z[3];
[ØØØ]  279:
[TR1]  28Ø:   z[Ø] = ESC;
[LM1]  281:   z[1] = 'w';
[781]  282:   z[2] = Ø;
[AN3]  283:   drucker_ausgabe(3,z);
[U7Ø]  284: }
[ØØØ]  285:
[7G3]  286: void prop_mode(void)
[P7Ø]  287: {
[CS1]  288:   char z[3];
[ØØØ]  289:
[TR1]  29Ø:   z[Ø] = ESC;
[5M1]  291:   z[1] = 'p';
[881]  292:   z[2] = 1;
[AN3]  293:   drucker_ausgabe(3,z);
[U7Ø]  294: }
[ØØØ]  295:
[S54]  296: void no_prop_mode(void)
[P7Ø]  297: {
[CS1]  298:   char z[3];
[ØØØ]  299:
[TR1]  3ØØ:   z[Ø] = ESC;
[5M1]  3Ø1:   z[1] = 'p';
[781]  3Ø2:   z[2] = Ø;
[AN3]  3Ø3:   drucker_ausgabe(3,z);
[U7Ø]  3Ø4: }
[ØØØ]  3Ø5:
[2E3]  3Ø6: void fett_mode(void)
[P7Ø]  3Ø7: {
[BS1]  3Ø8:   char z[2];
[ØØØ]  3Ø9:
[TR1]  31Ø:   z[Ø] = ESC;
[FL1]  311:   z[1] = 'E';
[8N3]  312:   drucker_ausgabe(2,z);
[U7Ø]  313: }
[ØØØ]  314:
[N44]  315: void no_fett_mode(void)
[P7Ø]  316: {
[BS1]  317:   char z[2];
[ØØØ]  318:
[TR1]  319:   z[Ø] = ESC;
[HL1]  32Ø:   z[1] = 'F';
[8N3]  321:   drucker_ausgabe(2,z);
[U7Ø]  322: }
[ØØØ]  323:
[KC4]  324: void benutz_z_satz(void)
[P7Ø]  325: {
[CS1]  326:   char z[3];
[ØØØ]  327:
[TR1]  328:   z[Ø] = ESC;
[FH1]  329:   z[1] = '%';
[881]  33Ø:   z[2] = 1;
[AN3]  331:   drucker_ausgabe(3,z);
[U7Ø]  332: }
```

Listing 2. Mit der Unit »drucker.c« wird die Druckerinitialisierung vorgenommen

```
[ENE]    1: /*_____________________ UNIT zur Maussteue
         rung _____________________
[ØØØ]    2:
[A74]    3:     Autor              : Markus Graf
[4W2]    4:     Version            : 1.2
[7H4]    5:     Zuerst erstellt    : 18.Ø3.199Ø
[6X4]    6:     Zuletzt geändert   : 25.Ø3.199Ø
[ØØØ]    7:
[ØØØ]    8:
[C44]    9:     Beschreibung Maus-UNIT:
[ØØØ]   1Ø:
[K77]   11:          int init_graph_mouse(int graphik_ka
         rte);
[F68]   12:               Maustreiber initialisieren, M
         auscursor = Pfeil,
[D67]   13:               graphik_karte = HERCULES oder
          NO_HERCULES,
[ØØ9]   14:               liefert -1 bei erfolgreicher
         Initialisierung, sonst Ø.
[ØØØ]   15:
[B44]   16:          void mouse_pfeil(void);
[G93]   17:               Mauscursor = Pfeil.
[ØØØ]   18:
[255]   19:          void mouse_pfeil_weiss(void);
```

```
[HT5]   2Ø:                 Mauscursor = Pfeil, weiß gefü
        llt.
[ØØØ]   21:
[3N3]   22:         void mouse_uhr(void);
[2W2]   23:                 Mauscursor = Uhr.
[ØØØ]   24:
[HF3]   25:         void mouse_on(void);
[536]   26:                 Mauscursor auf dem Bildschirm
         zeigen.
[ØØØ]   27:
[NL3]   28:         void mouse_off(void);
[ØD7]   29:                 Mauscursor nicht auf dem Bild
        schirm zeigen.
[ØØØ]   3Ø:
[PU6]   31:         void mouse_info(int *x,int *y,int *
        taste)
[H4A]   32:                 Die Funktion liefert die mome
        ntane Position und den Status
[BD8]   33:                 der linken Maustaste ( taste
        == 1 => Taste gedrückt)
[ØØØ]   34:
[KT8]   35:         mouse_reg mouse(int ax,int bx,int c
        x,int dx,int es);
[XK6]   36:                 übergibt Daten an den Mausint
        errupt 51,
[ET8]   37:                 liefert den Inhalt der Regist
        er AX,BX,CX,DX über den
[YH3]   38:                 Datentyp mouse_reg.
[ØØØ]   39:
[17Ø]   4Ø: */
[ØØØ]   41:
[ØØØ]   42:
[GØ9]   43: /********************** Variablen/Typ Defi
        nitionen *****************/
[ØØØ]   44:
[8M2]   45: typedef struct
[P7Ø]   46:   {
[ZN2]   47:     int ax,bx,cx,dx;
[X52]   48:   } mouse_reg;
[ØØØ]   49:
[ØØØ]   5Ø:
[ZV8]   51: /********************** Funktions-Prototyp
        en ***********************/
[ØØØ]   52:
[5T4]   53: int init_graph_mouse(int);
[B44]   54: void mouse_pfeil(void);
[255]   55: void mouse_pfeil_weiss(void);
[3N3]   56: void mouse_uhr(void);
[HF3]   57: void mouse_on(void);
[NL3]   58: void mouse_off(void);
[XR6]   59: mouse_reg mouse(int,int,int,int,int);
[ØØØ]   6Ø:
[ØØØ]   61:
[5X9]   62: /********************** Unterprogramme zur
         Maussteuerung ***********/
[ØØØ]   63:
[R57]   64: int init_graph_mouse(int graphik_karte)
[MØ5]   65: /* Maustreiber initialisieren;
[MTA]   66:    Wenn Initialisierung erfolgreich war, is
        t der Funktionswert -1,
[TW8]   67:    Bei fehlendem Maustreiber ist der Funkti
        onswert Ø;  */
[P7Ø]   68: {
[VZ6]   69:   if (graphik_karte == HERCMONO) pokeb(64,7
        3,6);
[3UA]   7Ø:   if (mouse(Ø,Ø,Ø,Ø,Ø).ax==Ø) return(Ø);  /
        * Maustreiber nicht vorhanden */
[VG2]   71:   mouse_pfeil();
[WT1]   72:   return(-1);
[U7Ø]   73: }
[ØØØ]   74:
[4U6]   75: /*.....................................
        ..............*/
[ØØØ]   76:
[ZW3]   77: void mouse_pfeil(void)
[AN3]   78: /* Mauscursor = Pfeil */
[P7Ø]   79: {
[KK9]   8Ø:   unsigned int pfeil[2][16]={ { Øx3fff,ØxØ3
        ff,ØxØØff,ØxØØ1f,
[6W3]   81:                                 ØxØØØf,ØxØØ
        1f,ØxØØ3f,ØxØØ7f,
[XW3]   82:                                 ØxØØ3f,ØxØØ
        1f,Øx0e0f,Øx3fØ7,
[1T4]   83:                                 Øxff83,Øxff
        c1,Øxffe3,Øxfff7 } ,
[A34]   84:                               { ØxØØØØ,Øx4Ø
        ØØ,Øx78ØØ,Øx4fØØ,
[PR3]   85:                                 Øx40eØ,Øx41
        cØ,Øx41ØØ,Øx41ØØ,
[ZU3]   86:                                 Øx5f8Ø,Øx71
        cØ,Øx40eØ,ØxØØ7Ø,
[KK4]   87:                                 ØxØØ38,ØxØØ
        1c,ØxØØØ8,ØxØØØØ } };
[PG6]   88:   mouse(9,Ø,Ø,FP_OFF(pfeil),FP_SEG(pfeil));
[U7Ø]   89: }
[ØØØ]   9Ø:
[4U6]   91: /*.....................................
        ..............*/
[ØØØ]   92:
[C15]   93: void mouse_pfeil_weiss(void)
[LN6]   94: /* Mauscursor = Pfeil, weiß gefüllt */
[P7Ø]   95: {
[KK9]   96:   unsigned int pfeil[2][16]={ { Øx3fff,ØxØ3
        ff,ØxØØff,ØxØØ1f,
[6W3]   97:                                 ØxØØØf,ØxØØ
        1f,ØxØØ3f,ØxØØ7f,
[XW3]   98:                                 ØxØØ3f,ØxØØ
        1f,Øx0e0f,Øx3fØ7,
[1T4]   99:                                 Øxff83,Øxff
        c1,Øxffe3,Øxfff7 } ,
[L34]   1ØØ:                              { ØxØØØØ,Øx4Ø
        ØØ,Øx78ØØ,Øx7fØØ,
[7Z3]   1Ø1:                                Øx7feØ,Øx7f
        cØ,Øx7fØØ,Øx7fØØ,
[7V3]   1Ø2:                                Øx7f8Ø,Øx71
        cØ,Øx40eØ,ØxØØ7Ø,
[KK4]   1Ø3:                                ØxØØ38,ØxØØ
        1c,ØxØØØ8,ØxØØØØ } };
[PG6]   1Ø4:   mouse(9,Ø,Ø,FP_OFF(pfeil),FP_SEG(pfeil));
[U7Ø]   1Ø5: }
[ØØØ]   1Ø6:
[4U6]   1Ø7: /*.....................................
        ..............*/
[ØØØ]   1Ø8:
[DH3]   1Ø9: void mouse_uhr(void)
[RE3]   11Ø: /* Mauscursor = Uhr */
[P7Ø]   111: {
[D39]   112:   unsigned int uhr[2][16]={ { Øxf81f,Øxf ØØf
        ,ØxeØØ7,ØxcØØ3,
[BN3]   113:                                Øx8ØØ1,ØxØØØØ
        ,ØxØØØØ,ØxØØØØ,
[MM3]   114:                                ØxØØØØ,ØxØØØØ
        ,ØxØØØØ,Øx8ØØ1,
[BF4]   115:                                ØxcØØ3,ØxeØØ7
        ,Øxf ØØf,Øxf81f } ,
[TA4]   116:                              {  ØxØØØØ,Øx07e
        Ø,Øx0cbØ,Øx1898,
[UW3]   117:                                Øx3ØØc,Øx64Ø6
        ,Øx42Ø2,Øx71fa,
[TU3]   118:                                Øx418e,Øx4ØØ2
        ,Øx6ØØ6,Øx3ØØc,
[8R4]   119:                                Øx1918,Øx0d3Ø
        ,Øx07eØ,ØxØØØØ } };
[ØØØ]   12Ø:
[8V5]   121:   mouse(9,Ø,Ø,FP_OFF(uhr),FP_SEG(uhr));
[U7Ø]   122: }
[ØØØ]   123:
[4U6]   124: /*.....................................
        ..............*/
[ØØØ]   125:
[PD3]   126: void mouse_on(void)
[CG4]   127: /* Mauscursor einschalten */
[P7Ø]   128: {
[L92]   129:   mouse(1,Ø,Ø,Ø,Ø);
[U7Ø]   13Ø: }
[ØØØ]   131:
[4U6]   132: /*.....................................
        ..............*/
[ØØØ]   133:
[ZF3]   134: void mouse_off(void)
[KG4]   135: /* Mauscursor ausschalten */
[P7Ø]   136: {
[R92]   137:   mouse(2,Ø,Ø,Ø,Ø);
[U7Ø]   138: }
[ØØØ]   139:
[4U6]   14Ø: /*.....................................
        ..............*/
[ØØØ]   141:
[PU6]   142: void mouse_info(int *x,int *y,int *taste)
[9ØA]   143: /* Die Funktion liefert die momentane Posit
        ion und den Status
[5T8]   144:    der linken Maustaste ( taste == 1 => Tas
        te gedrückt) */
[P7Ø]   145: {
[CA2]   146:   mouse_reg mr;
[ØØØ]   147:
[ØZ2]   148:   mr = mouse(3,Ø,Ø,Ø,Ø);
[GT1]   149:   *x = mr.cx;
[NT1]   15Ø:   *y = mr.dx;
[9V2]   151:   *taste = mr.bx & 1;
[U7Ø]   152: }
[ØØØ]   153:
[4U6]   154: /*.....................................
        ..............*/
[ØØØ]   155:
[RR8]   156: mouse_reg mouse(int ax,int bx,int cx,int dx
        ,int es)
[FØ9]   157: /* Die Funktion übergibt Daten an den Mausi
        nterrupt 51,
[KZ8]   158:    Zurückgegeben wird der Inhalt der Regist
        er AX,BX,CX,DX
[645]   159:    über den Datentyp mouse_reg.*/
[P7Ø]   16Ø: {
[MH2]   161:   union REGS reg;
```

Listing 3. Alle Maussteuerungsfunktionen sind in der Unit »mouse.c« verankert

```
[FD3]   162:   struct SREGS sregs;
[CA2]   163:   mouse_reg mr;
[ØØØ]   164:
[B12]   165:   reg.x.ax=ax;
[K12]   166:   reg.x.bx=bx;
[T12]   167:   reg.x.cx=cx;
[Ø22]   168:   reg.x.dx=dx;
[Y22]   169:   sregs.es=es;
[5A4]   17Ø:   int86x(51,&reg,&reg,&sregs);
[TM2]   171:   mr.ax=reg.x.ax;
[WM2]   172:   mr.bx=reg.x.bx;
[ZM2]   173:   mr.cx=reg.x.cx;
[2N2]   174:   mr.dx=reg.x.dx;
[YW1]   175:   return(mr);
[U7Ø]   176: }
```

Listing 3. Alle Maussteuerungsfunktionen sind in der Unit »mouse.c« verankert

```
[ØGD]     1: /*_____________ UNIT zur Graphikausgabe
            & Dateiverwaltung _________
[ØØØ]     2:
[A74]     3:     Autor              : Markus Graf
[4W2]     4:     Version            : 1.2
[SM4]     5:     Zuerst erstellt    : 18.Ø3.199Ø
[1X4]     6:     Zuletzt geändert   : 14.Ø3.199Ø
[ØØØ]     7:
[ØØØ]     8:
[H35]     9:     Beschreibung Dateifunktionen:
[ØØØ]     1Ø:
[9W7]    11:          int handler(int errval,int rax,int
            rbp,int rsi);
[D89]    12:                  MS/DOS Fehlermeldungen abfa
            ngen und Error-Box ausgeben.
[YAA]    13:                  Bei Disketten-Fehler wird M
            S/DOS-Funktion abgebrochen und zum
[5L4]    14:                  C-Programm zurückgekehrt.
[21B]    15:                  Bei anderen Fehlern wird in
             den Textmodus zurückgeschaltet und
[3K3]    16:                  das Programm beendet.
[ØX8]    17:                  Die Funktion wird mit    ha
            rderr(handler);   aktiviert.
[ØØØ]    18:
[Ø3A]    19:         char *datei_box(int x1,int y1,char
            *suchparameter,char *titel,
[WZ3]    2Ø:                           int color,int h_col
            or);
[CR7]    21:                  Gibt eine Datei-Box an der
            Position (x1,y1) aus.
[CM6]    22:                  suchparameter : enthällt Su
            chpattern,
[RM4]    23:                  titel : Titel der Datei-Box
            ,
[979]    24:                  datei : es kann ein Pfad mi
            t Dateiname vorgegeben werden,
[5M9]    25:                  Als Verzeichnis wird das ak
            tuelle Verzeichnis genommen,
[ZY8]    26:                  Als Ergebnis wird der gewäh
            lte Dateipfad übergeben.
[ØØØ]    27:
[ØØØ]    28:
[3P5]    29:     Beschreibung Graphikfunktionen:
[ØØØ]    3Ø:
[BS4]    31:          void error_box(char *text);
[ZK9]    32:                  Fehlermeldung ausgeben und
            warten bis "OK" angeklickt wurde.
[ØØØ]    33:
[EV5]    34:          int box(int x1,int y1,int x2,int y2
            ,
[1U3]    35:                           char *titel,char *tex
            t,
[A46]    36:                           char *knopf1,char *kn
            opf2,char *knopf3,
[WZ3]    37:                           int color,int h_color
            );
[53A]    38:                  Box an den gegebenen Koordi
            naten mit Titel und Text ausgeben;
[LTA]    39:                  knopf1,knopf2 und knopf3 en
            thalten die Beschriftung der Knöpfe,
[WR7]    4Ø:                  knopfn == "" => dieser Knop
            f wird nicht benutzt.
[US9]    41:                  Ein Knopf, der in eckigen K
            lammern geschrieben ist, kann auch
[7P4]    42:                  mit RETURN gewählt werden.
[XY7]    43:                  Zurückgegeben wird die gewä
            hlte Knopfnummer.
[ØØØ]    44:
[WCA]    45:          void putbox(int x1,int y1,int x2,in
            t y2,char *titel,char *text,
[WZ3]    46:                           int color,int h_color);
[B2A]    47:                  Box an den gegebenen Koordi
            naten mit Titel und Text ausgeben.
[FR9]    48:                  Enthält text das '|' Zeiche
            n, wird der Text hinter diesem
[GV6]    49:                  Zeichen in der nächsten Zei
            le ausgegeben.
[ØØØ]    5Ø:
[G19]    51:          void set_knopf(int x1,int y1,int x2
            ,int y2,char *text);
[2V9]    52:                  Gibt einen Knopf an den geg
            ebenen Koordinaten mit Text aus.
[ØØØ]    53:
[5G7]    54:          void invertimage(int x1,int y1,int
            x2,int y2);
[TL6]    55:                  Invertiert einen Bildschirm
            ausschnitt
[ØØØ]    56:
[SH9]    57:          void far *getimage_in_puffer(int x1
            ,int y1,int x2,int y2);
[ZD9]    58:                  Reserviert Speicherplatz, i
            n den dann der Bildausschnitt
[GP9]    59:                  kopiert wird. Zurückgeliefe
            rt wird die Speicheradresse.
[17Ø]    6Ø: */
[ØØØ]    61:
[3Ø9]    62: /********************** Konstanten Definit
            ionen ********************/
[ØØØ]    63:
[183]    64: #define  DRIVE_A     Ø
[PC3]    65: #define  DRIVE_B      1
[B43]    66: #define DRIVE_C      2
[ØY2]    67: #define FERTIG       3
[TS3]    68: #define ABBRUCH      Øx8ØØØ
[RP2]    69: #define HOCH         5
[N73]    7Ø: #define  RUNTER       6
[YK3]    71: #define  DATEI_EDIT   7
[HX4]    72: #define MAX_VERZEICHNISSE  3ØØ
[ØØØ]    73:
[H23]    74: #define  RETURN  13
[PD2]    75: #define  BS      8
[TM2]    76: #define  ESC      27
[L33]    77: #define  DEL      Øx53ØØ
[XP3]    78: #define  RECHTS  Øx4DØØ
[RD3]    79: #define  LINKS    Øx4BØØ
[ØØØ]    8Ø:
[ØØØ]    81:
[GØ9]    82: /********************** Variablen/Typ Defi
            nitionen *****************/
[ØØØ]    83:
[161]    84: struct
[P7Ø]    85:   {
[FK4]    86:     char name[MAXFILE+MAXEXT];
[Y3C]    87:   } file[MAX_VERZEICHNISSE];          /* Sp
            eicher für Inhaltsverzeichnis */
[ØØØ]    88:
[SN9]    89: int druck_err = OK;                    /* F
            ehler, wenn Druckerausgabe
[S12]    9Ø:                                         Fe
            hlerhaft */
[ØØØ]    91:
[ZV8]    92: /********************** Funktions-Prototyp
            en **********************/
[ØØØ]    93:
[FX4]    94: int    handler(int,int,int,int);
[UG8]    95: char  *datei_box(int,int,char *,char *,char
             *,int,int);
[L86]    96: int    get_name(int,int,char *,char *);
[ZV3]    97: int    get_taste(void);
[3R5]    98: void  cursor(int,int,int,int);
[KM6]    99: void  inv_datei_knopf(int,int,int,int);
[3M6]   1ØØ: int    neuer_pfad(int,int,char *,int,int);
[8U5]   1Ø1: void  file_liste(int,int,int,int);
[LL4]   1Ø2: int    get_directory(char *);
[9W3]   1Ø3: void  error_box(char *);
[NMA]   1Ø4: int    box(int,int,int,int,char *,char *,ch
            ar *,char *,char *,int,int);
[T78]   1Ø5: void  putbox(int,int,int,int,char *,char *,
            int,int);
[EC3]   1Ø6: char  get_key(char *);
[GV6]   1Ø7: void  set_knopf(int,int,int,int,char *);
[BU5]   1Ø8: void  invertimage(int,int,int,int);
[6U7]   1Ø9: void  far *getimage_in_puffer(int,int,int,i
            nt);
[ØØØ]   11Ø:
[ØØØ]   111:
[L2A]   112: /********************** Unterprogramme zur
             Dateiverwaltung *********/
[ØØØ]   113:
[EU7]   114: int handler(int errval,int rax,int rbp,int
            rsi)
[ØØØ]   115:
[699]   116: /* MS/DOS Fehlermeldungen abfangen und Erro
            r-Box ausgeben.
[YAA]   117:    Bei Disketten-Fehler wird MS/DOS-Funktio
            n abgebrochen und zum
[5L4]   118:    C-Programm zurückgekehrt.
[21B]   119:    Bei anderen Fehlern wird in den Textmodu
            s zurückgeschaltet und
[3K3]   12Ø:    das Programm beendet.
[159]   121:    Die Funktion wird mit    harderr(handler
```

```
            );   aktiviert. */
[P7Ø]   122: {
[VK7]   123:     char text[35];     /* String Zwischenspe
            icher */
[ØØØ]   124:
[VG2]   125:   mouse_pfeil();
[CF1]   126:   if (rax>Ø)
[P7Ø]   127:   {
[CW6]   128:     strcpy(text,"     Fehler in Laufwerk
            :     ");
[F76]   129:     text[strlen(text)-7]=(rax & Øxff) + 'A'
            ;
[RV2]   13Ø:     error_box(text);
[A12]   131:     hardretn(-1);
[F11]   132:   } else
[P7Ø]   133:   {
[V4G]   134:     error_box("Fehler bei der Datenuebertra
            gung auf den Drucker !|Druckerausgabe wird
            abgebrochen");
[K53]   135:     druck_err = ERROR;
[A12]   136:     hardretn(-1);
[U7Ø]   137:   }
[U7Ø]   138: }
[ØØØ]   139:
[4U6]   14Ø: /*.....................................
            ...............*/
[ØØØ]   141:
[CAA]   142: char *datei_box(int x1,int y1,
                /* Koordinaten der Box  */
[RD7]   143:            char *suchparameter,
                /* z.B: "*.FNT"         */
[YU7]   144:            char *titel,
                /* Überschrift der Box  */
[3M7]   145:            char *datei,
                /* event. Dateivorgabe  */
[CK7]   146:            int color,
                /* Vordergrundfarbe     */
[AV7]   147:            int h_color)
                /* Hintergrundfarbe     */
[ØØØ]   148:
[6R7]   149: /* Gibt eine Datei-Box an der Position (x1,
            y1) aus.
[5M9]   15Ø:    Als Verzeichnis wird das aktuelle Verzei
            chnis genommen,
[Ø79]   151:    Als Ergebnis wird der gewählte Dateipfad
             übergeben. */
[P7Ø]   152: {
[XT7]   153:   void *picptr;                  /* Bildzwi
            schenspeicher */
[NG6]   154:   char pfad[MAXPATH];            /* Dateipf
            ad */
[APA]   155:   int files,                     /* Anzahl d
            er Dateien in einem Verzeichnis */
[DL4]   156:       drive,                     /* Laufwerk
             */
[385]   157:       x,y,                       /* Mausposi
            tion */
[RT4]   158:       taste,                     /* Maustast
            e */
[MW6]   159:       anfang,                    /* Anfang
            der Dateiliste */
[976]   16Ø:       datei_nr,                  /* gewählt
            e Datei */
[HU7]   161:       datei_nr_alt,              /* vorher
            gewählte Datei */
[PV8]   162:       laufwerke;                 /* Anzahl d
            er Diskettenlaufwerke */
[GP6]   163:   long wahl;                     /* gewählte
            r Knopf */
[TU9]   164:   char edit_name[MAXFILE+MAXEXT];  /* editi
            erter Dateiname */
[161]   165:   struct
[P7Ø]   166:   {
[YD3]   167:     char name[MAXFILE];
[513]   168:     char ext[MAXEXT];
[7A5]   169:   } edit;                         /* Dateina
            me */
[KH8]   17Ø:   union REGS regs;               /* Variable
            n für Interrupt */
[ØØØ]   171:
[U12]   172:   mouse_off();
[HR7]   173:   picptr = getimage_in_puffer(x1,y1,x1+226,
            y1+2ØØ);
[4X7]   174:   putbox(x1,y1,x1+226,y1+2ØØ,titel,"",color
            ,h_color);
[SA6]   175:   set_knopf(x1+141,y1+7Ø,x1+217,y1+82,"(O)K
            ");
[Ø77]   176:   set_knopf(x1+141,y1+86,x1+217,y1+98,"(A)b
            bruch");
[236]   177:   set_knopf(x1+147,y1+13Ø,x1+182,y1+142,"A:
            ");
[AL3]   178:   int86(Øx11,&regs,&regs);
[9Ø3]   179:   if (regs.x.ax & ØxCØ)
[P7Ø]   18Ø:   {
[B46]   181:     set_knopf(x1+147,y1+146,x1+182,y1+158,"
            B:");
[W36]   182:     set_knopf(x1+147,y1+162,x1+182,y1+174,"
            C:");
[S82]   183:     laufwerke = 2;
[F11]   184:   } else
[P7Ø]   185:   {
[C46]   186:     set_knopf(x1+147,y1+146,x1+182,y1+158,"
            C:");
[P82]   187:     laufwerke = 1;
[U7Ø]   188:   }
[E47]   189:   settextstyle(DEFAULT_FONT,HORIZ_DIR,2);
[ØR5]   19Ø:   set_knopf(x1+116,y1+62,x1+137,y1+82,"");
[TT5]   191:   set_knopf(x1+116,y1+86,x1+137,y1+1Ø6,"");
[GM4]   192:   outtextxy(x1+119,y1+62,"\36");
[4N4]   193:   outtextxy(x1+119,y1+86,"\37");
[C47]   194:   settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
[CT2]   195:   setcolor(color);
[LU5]   196:   outtextxy(x1+132,y1+118,"Laufwerk:");
[9Y2]   197:   drive = getdisk();
[8V6]   198:   inv_datei_knopf(x1,y1,drive,laufwerke);
[KM2]   199:   if (datei[Ø] == Ø)
[P7Ø]   2ØØ:   {
[CC3]   2Ø1:     strcpy(edit_name,"");
[G62]   2Ø2:     datei_nr = -1;
[F11]   2Ø3:   } else
[P7Ø]   2Ø4:   {
[5E7]   2Ø5:     fnsplit(datei,NULL,NULL,edit.name,edit.
            ext);
[US4]   2Ø6:     strcpy(edit_name,edit.name);
[7N4]   2Ø7:     strcat(edit_name,edit.ext);
[K62]   2Ø8:     datei_nr = -2;
[U7Ø]   2Ø9:   }
[D33]   21Ø:   if (edit_name[Ø]!=Ø)
[P87]   211:     set_knopf(x1+114,y1+36,x1+221,y1+48,edi
            t_name);
[ZRØ]   212:   else
[T58]   213:     set_knopf(x1+114,y1+36,x1+221,y1+48,"xx
            xxxxxx.xxx");
[P89]   214:   files = neuer_pfad(x1,y1,suchparameter,co
            lor,h_color);
[KN1]   215:   anfang = 1;
[Z64]   216:   datei_nr_alt = datei_nr;
[PØ2]   217:   mouse_on();
[NBØ]   218:   do
[P7Ø]   219:   {
[MC1]   22Ø:     wahl = -1;
[NBØ]   221:     do
[P7Ø]   222:     {
[V24]   223:       mouse_info(&x,&y,&taste);
[YN1]   224:       if (kbhit())
[P7Ø]   225:       {
[XV4]   226:         taste = toupper(get_taste());
[WS4]   227:         if (taste == 'O') wahl = FERTIG;
[CU4]   228:         if (taste == 'A') wahl = ABBRUCH;
[U7Ø]   229:       }
[RY4]   23Ø:     } while((taste==Ø) && (wahl==-1));
[8N8]   231:     if ((x>x1+114) && (x<x1+221) && (y>y1+3
            6) && (y<y1+48) && (taste==1))
[P7Ø]   232:     {
[U12]   233:       mouse_off();
[8Ø8]   234:       if (get_name(x1,y1,edit_name,suchpara
            meter)==OK)
[P7Ø]   235:       {
[1K2]   236:         if (datei_nr>=Ø)
[P7Ø]   237:         {
[FWC]   238:           invertimage(x1+8,y1+33+(datei_nr-
            anfang+1)*12,x1+1Ø8,y1+43+(datei_nr-anfang+
            1)*12);
[K62]   239:           datei_nr = -2;
[U7Ø]   24Ø:         }
[466]   241:         if (strlen(edit_name)==Ø) datei_nr
            = -1;
[643]   242:         else datei_nr = -2;
[U7Ø]   243:       }
[PØ2]   244:       mouse_on();
[Z64]   245:       datei_nr_alt = datei_nr;
[U7Ø]   246:     }
[RAB]   247:     if ((x>x1+147) && (x<x1+182) && (taste=
            =1))          /* neues Laufwerk gewählt ? *
            /
[P7Ø]   248:     {
[476]   249:       if ((y>y1+13Ø) && (y<y1+142)) wahl =
            DRIVE_A;
[LM3]   25Ø:       if ((y>y1+146) && (y<y1+158))
[P7Ø]   251:       {
[AF5]   252:         if (laufwerke == 1) wahl = DRIVE_C;
[YB3]   253:         else wahl = DRIVE_B;
[U7Ø]   254:       }
[ZV8]   255:       if ((y>y1+162) && (y<y1+174) && (lauf
            werke==2)) wahl = DRIVE_C;
[FG9]   256:       if ((wahl==DRIVE_A) || (wahl==DRIVE_B
            ) || (wahl==DRIVE_C))
[P7Ø]   257:       {
[8E2]   258:         setdisk(wahl);
[3B3]   259:         if (getdisk() == wahl)
[P7Ø]   26Ø:         {
[8V6]   261:           inv_datei_knopf(x1,y1,drive,laufw
            erke);
[922]   262:           drive = wahl;
```

Listing 4. »grafik.c« enthält eine Unit, die die Grafikausgabe und Dateiverwaltung übernimmt

```
[8V6]  263:           inv_datei_knopf(x1,y1,drive,laufw
       erke);
[U7Ø]  264:         }
[P89]  265:         files = neuer_pfad(x1,y1,suchparame
       ter,color,h_color);
[KN1]  266:         anfang = 1;
[U7Ø]  267:       }
[U7Ø]  268:     }
[2L5]  269:     if ((x>x1+116) && (x<x1+137) && (taste=
       =1))
[P7Ø]  27Ø:     {
[TG3]  271:       if ((y>y1+62) && (y<y1+82))
[P7Ø]  272:       {
[YU1]  273:         wahl = HOCH;
[135]  274:         anfang = (anfang>1Ø)? anfang-1Ø : 1
       ;
[N85]  275:         file_liste(x1,y1,anfang,files);
[GV2]  276:         if (datei_nr != -2)
[P7Ø]  277:         {
[9S4]  278:           datei_nr = datei_nr_alt = -1;
[T58]  279:           set_knopf(x1+114,y1+36,x1+221,y1+
       48,"xxxxxxxx.xxx");
[U7Ø]  28Ø:         }
[U7Ø]  281:       }
[AL3]  282:       if ((y>y1+86) && (y<y1+1Ø6))
[P7Ø]  283:       {
[492]  284:         wahl = RUNTER;
[1EA]  285:         if (files > 1Ø)  anfang = ((anfang+
       1Ø)<(files-1Ø))? anfang+1Ø : files-1Ø;
[TY3]  286:         if (files <= 1Ø) anfang = 1;
[N85]  287:         file_liste(x1,y1,anfang,files);
[GV2]  288:         if (datei_nr != -2)
[P7Ø]  289:         {
[9S4]  29Ø:           datei_nr = datei_nr_alt = -1;
[T58]  291:           set_knopf(x1+114,y1+36,x1+221,y1+
       48,"xxxxxxxx.xxx");
[U7Ø]  292:         }
[U7Ø]  293:       }
[U7Ø]  294:     }
[RK5]  295:     if ((x>x1+141) && (x<x1+217) && (taste=
       =1))
[P7Ø]  296:     {
[DT5]  297:       if ((y>y1+7Ø) && (y<y1+82)) wahl = FE
       RTIG;
[SØ6]  298:       if ((y>y1+86) && (y<y1+98)) wahl = AB
       BRUCH;
[U7Ø]  299:     }
[LP8]  3ØØ:     if ((x>x1+7) && (x<x1+1Ø9) && (y>y1+32)
        && (y<y1+188) && (taste==1))
[P7Ø]  3Ø1:     {
[ZY4]  3Ø2:       datei_nr = (y-y1-32)/12+anfang-1;
[E83]  3Ø3:       if (datei_nr < files)
[P7Ø]  3Ø4:       {
[1L5]  3Ø5:         if (file[datei_nr].name[Ø] == '\\')
[P7Ø]  3Ø6:         {
[LC5]  3Ø7:           chdir(&file[datei_nr].name[1]);
[P89]  3Ø8:           files = neuer_pfad(x1,y1,suchpara
       meter,color,h_color);
[KN1]  3Ø9:           anfang = 1;
[9S4]  31Ø:           datei_nr = datei_nr_alt = -1;
[F11]  311:         } else
[P7Ø]  312:         {
[U12]  313:           mouse_off();
[GR4]  314:           if (datei_nr!=datei_nr_alt)
[P7Ø]  315:           {
[PK2]  316:             if (datei_nr!=-2)
[FWC]  317:               invertimage(x1+8,y1+33+(datei
       _nr-anfang+1)*12,x1+1Ø8,y1+43+(datei_nr-anf
       ang+1)*12);
[4U6]  318:             if ((datei_nr_alt!=-1) && (date
       i_nr_alt!=-2))
[TCE]  319:               invertimage(x1+8,y1+33+(datei
       _nr_alt-anfang+1)*12,x1+1Ø8,y1+43+(datei_nr
       _alt-anfang+1)*12);
[U7Ø]  32Ø:           }
[Z64]  321:           datei_nr_alt = datei_nr;
[65A]  322:           fnsplit(file[datei_nr].name,NULL,
       NULL,edit.name,edit.ext);
[US4]  323:           strcpy(edit_name,edit.name);
[7N4]  324:           strcat(edit_name,edit.ext);
[P87]  325:           set_knopf(x1+114,y1+36,x1+221,y1+
       48,edit_name);
[PØ2]  326:           mouse_on();
[U7Ø]  327:         }
[L83]  328:       } else datei_nr = -1;
[U7Ø]  329:     }
[NBØ]  33Ø:     do
[P7Ø]  331:     {
[V24]  332:       mouse_info(&x,&y,&taste);
[FY2]  333:     } while(taste!=Ø);
[UP9]  334:   } while((wahl!=ABBRUCH) && (wahl!=FERTIG
       || datei_nr==-1));
[U12]  335:   mouse_off();
[HF5]  336:   putimage(x1,y1,picptr,COPY_PUT);
[PØ2]  337:   mouse_on();
[GC5]  338:   if ((datei_nr!=-1) && (wahl==FERTIG))
[P7Ø]  339:   {
[WK3]  34Ø:     pfad[Ø]='A'+getdisk();
[7W1]  341:     pfad[1]=':';
```

```
[692]  342:     pfad[2]='\\';
[TS1]  343:     pfad[2Ø]=Ø;
[BK3]  344:     getcurdir(Ø,&pfad[3]);
[UØ6]  345:     if (strlen(pfad)>3) strcat(pfad,"\\");
[8F8]  346:     if (datei_nr!=-2) strcat(pfad,file[date
       i_nr].name);
[VU4]  347:     else strcat(pfad,edit_name);
[AP2]  348:   } else pfad[Ø]=Ø;
[ØR2]  349:   farfree(picptr);
[M42]  35Ø:   return(pfad);
[U7Ø]  351: }
[ØØØ]  352:
[4U6]  353: /*.....................................
       ..............*/
[ØØØ]  354:
[3F8]  355: int get_name(int x1,int y1,            /* K
       oordinaten      */
[M76]  356:              char *name,               /* e
       ditierter Text  */
[LL9]  357:              char *suchparameter)      /* z.
       B.: "*.FNT" für evntl. Endung zu
[GX1]  358:
                        ergänzen */
[HK3]  359: /* Dateiname eingeben */
[P7Ø]  36Ø: {
[S19]  361:   char text_buffer[MAXFILE+MAXEXT],    /* T
       extspeicher */
[M6A]  362:       text_z_buffer[MAXFILE+MAXEXT];  /* Te
       xtzwischenspeicher */
[NZ6]  363:   int  pos,                           /* S
       tringposition */
[135]  364:       key,                            /* Ta
       statur */
[FT5]  365:       lang,                           /* S
       tringlänge */
[ZN5]  366:       x,y,                            /* Bi
       ldposition */
[KL7]  367:       x_alt;                          /* vo
       rherige Bildposition */
[ØØØ]  368:
[9A2]  369:   x_alt = x1+116;
[DB1]  37Ø:   y = y1+39;
[E61]  371:   pos = Ø;
[BZ3]  372:   strcpy(text_buffer,"");
[VM5]  373:   setfillstyle(SOLID_FILL,BLACK);
[W74]  374:   bar(x1+116,y1+39,x1+211,y1+47);
[NBØ]  375:   do
[P7Ø]  376:   {
[1G2]  377:     x = pos*8+116+x1;
[VM5]  378:     setfillstyle(SOLID_FILL,BLACK);
[P74]  379:     bar(x1+115,y1+37,x1+22Ø,y1+47);
[CT2]  38Ø:     setcolor(WHITE);
[1G5]  381:     outtextxy(x1+116,y,text_buffer);
[YZ4]  382:     cursor(x_alt,x,y,WHITE,BLACK);
[UM1]  383:     x_alt = x;
[V23]  384:     key = get_taste();
[5X1]  385:     switch(key)
[P7Ø]  386:     {
[AS4]  387:       case LINKS:  if (pos>Ø) pos--;
[PØ1]  388:                    break;
[AP9]  389:       case RECHTS:if ((pos<strlen(text_buff
       er)) && (pos<11)) pos++;
[PØ1]  39Ø:                  break;
[453]  391:       case BS:     if (pos>Ø)
[P7Ø]  392:                    {
[HØ1]  393:                      pos--;
[435]  394:                      if (pos<=strlen(text_bu
       ffer))
[CA8]  395:                        strcpy(&text_buffer[p
       os],&text_buffer[pos+1]);
[ZRØ]  396:                      else
[UM3]  397:                        text_buffer[pos] = Ø;
[U7Ø]  398:                    }
[XS6]  399:       case DEL:    if (pos<strlen(text_buff
       er))
[CA8]  4ØØ:                    strcpy(&text_buffer[pos
       ],&text_buffer[pos+1]);
[PØ1]  4Ø1:                  break;
[U7Ø]  4Ø2:     }
[3S2]  4Ø3:     if (isgraph(key))
[P7Ø]  4Ø4:     {
[EN4]  4Ø5:       lang = strlen(text_buffer);
[2R1]  4Ø6:       if (lang<12)
[P7Ø]  4Ø7:       {
[856]  4Ø8:         strcpy(text_z_buffer,text_buffer);
[6X8]  4Ø9:         strcpy(&text_buffer[pos+1],&text_z_
       buffer[pos]);
[Ø64]  41Ø:         text_buffer[pos] = key;
[BØ1]  411:         pos++;
[L34]  412:         text_buffer[lang+1] = Ø;
[U7Ø]  413:       }
[U7Ø]  414:     }
[AN5]  415:   } while((key!=RETURN) && (key!=ESC));
[2N6]  416:   if (key!=ESC) strcpy(name,text_buffer);
[36B]  417:   if ((strchr(name,'.')==NULL) && (strchr(&
       suchparameter[1],'*')==NULL) &&
[FR2]  418:       (strlen(name)<9))
[5F6]  419:     strcat(name,strchr(suchparameter,'.'));
[Z62]  42Ø:   strupr(name);
```

```
[T96]  421:   set_knopf(x1+114,y1+36,x1+221,y1+48,name)
              ;
[MP4]  422:   if (key == ESC) return(ERROR);
[KW1]  423:   return(OK);
[U7Ø]  424: }
[ØØØ]  425:
[4U6]  426: /*.....................................
              ..............*/
[ØØØ]  427:
[ED3]  428: int get_taste(void)
[ØØØ]  429:
[P43]  43Ø: /* Tastatur lesen */
[P7Ø]  431: {
[235]  432:   int key;  /* Tastaturzeichen */
[ØØØ]  433:
[5N2]  434:   key = bioskey(Ø);
[L55]  435:   if ((key & ØxØØff) != Ø) key &= ØxØØff;
[5Ø2]  436:   return(key);
[U7Ø]  437: }
[ØØØ]  438:
[4U6]  439: /*.....................................
              ..............*/
[ØØØ]  44Ø:
[YMA]  441: void cursor(int x_alt,
                    /* alte Cursor X-Position */
[WE7]  442:              int x,int y,
                    /* neue Koordinaten */
[CK7]  443:              int color,
                    /* Vordergrundfarbe  */
[TT7]  444:              int h_color)
                    /* Hintergrundfarbe */
[ØØØ]  445:
[2V3]  446: /* Textcursor ausgeben,
[GØC]  447:    Cursor an der Position x_alt löschen und
               an der Position x ausgeben */
[P7Ø]  448: {
[336]  449:   setfillstyle(SOLID_FILL,h_color);
[RL4]  45Ø:   bar(x_alt,y+7,x_alt+7,y+7);
[CT2]  451:   setcolor(color);
[T73]  452:   outtextxy(x,y,"_");
[U7Ø]  453: }
[ØØØ]  454:
[4U6]  455: /*.....................................
              ..............*/
[ØØØ]  456:
[9SB]  457: void inv_datei_knopf(int x1,int y1,     /* l
            inke obere Ecke der Dateibox */
[P35]  458:                     int knopf,        /* Kn
            opfnummer */
[CL8]  459:                     int laufwerke)    /* An
            zahl der Diskettenlaufwerke */
[ØØØ]  46Ø:
[D65]  461: /* Invertiert Laufwerksknopf */
[P7Ø]  462: {
[U12]  463:   mouse_off();
[D62]  464:   switch(knopf)
[P7Ø]  465:   {
[6C8]  466:     case DRIVE_A: invertimage(x1+146,y1+129
            ,x1+183,y1+143);
[PØ1]  467:                   break;
[AC8]  468:     case DRIVE_B: invertimage(x1+146,y1+145
            ,x1+183,y1+159);
[PØ1]  469:                   break;
[F75]  47Ø:     case DRIVE_C:  if (laufwerke == 1)
[WØ6]  471:                     invertimage(x1+146,y1+1
            45,x1+183,y1+159);
[ZRØ]  472:                   else
[CØ6]  473:                     invertimage(x1+146,y1+1
            61,x1+183,y1+175);
[PØ1]  474:                   break;
[U7Ø]  475:   }
[PØ2]  476:   mouse_on();
[U7Ø]  477: }
[ØØØ]  478:
[4U6]  479: /*.....................................
              ..............*/
[ØØØ]  48Ø:
[F4B]  481: int neuer_pfad(int x1,int y1,           /*
             linke obere Ecke der Dateibox */
[256]  482:                char *suchparameter,    /*
            z.B.: "*.fnt"  */
[RA6]  483:                int color,              /*
            Vordergrundfarbe */
[NN6]  484:                int h_color)            /*
            Hintergrundfarbe */
[ØØØ]  485:
[ØY9]  486: /* Dateiliste des neuen aktuellen Verzeichn
            isses ausgeben */
[P7Ø]  487: {
[7B6]  488:   int files;              /* Anzahl der Dat
            eien */
[1A8]  489:   char  pfad[MAXPATH];    /* momentanes Ver
            zeichnis */
[ØØØ]  49Ø:
[C32]  491:   mouse_uhr();
[PØ2]  492:   mouse_on();
[AF6]  493:   files = get_directory(suchparameter);
[WK3]  494:   pfad[Ø]='A'+getdisk();
[7W1]  495:   pfad[1]=':';
```

```
[692]  496:   pfad[2]='\\';
[BP1]  497:   pfad[3]=Ø;
[BK3]  498:   getcurdir(Ø,&pfad[3]);
[UØ6]  499:   if (strlen(pfad)>3) strcat(pfad,"\\");
[MS4]  5ØØ:   strcat(pfad,suchparameter);
[CT2]  5Ø1:   setcolor(color);
[336]  5Ø2:   setfillstyle(SOLID_FILL,h_color);
[U12]  5Ø3:   mouse_off();
[UY3]  5Ø4:   bar(x1+5,y1+21,x1+223,y1+31);
[NT1]  5Ø5:   pfad[27]=Ø;
[ØD4]  5Ø6:   outtextxy(x1+5,y1+21,pfad);
[Ø54]  5Ø7:   file_liste(x1,y1,1,files);
[VG2]  5Ø8:   mouse_pfeil();
[PØ2]  5Ø9:   mouse_on();
[CB2]  51Ø:   return(files);
[U7Ø]  511: }
[ØØØ]  512:
[4U6]  513: /*.....................................
              ..............*/
[ØØØ]  514:
[7YA]  515: void file_liste(int x1,int y1,      /* link
            e obere Ecke der Dateibox */
[WY5]  516:                 int anfang,          /* Nr.
             der 1. Datei    */
[FD6]  517:                 int anzahl)          /* Anz
            ahl der Dateien  */
[ØØØ]  518:
[ØD5]  519: /* Dateien in der DateiBox listen */
[P7Ø]  52Ø: {
[9X4]  521:   int i;      /* Schleifenzähler */
[ØØØ]  522:
[YP5]  523:   setfillstyle(SOLID_FILL,WHITE);
[T44]  524:   bar(x1+7,y1+32,x1+1Ø9,y1+188);
[5R2]  525:   setcolor(BLACK);
[945]  526:   rectangle(x1+7,y1+32,x1+1Ø9,y1+188);
[M77]  527:   for(i=anfang-1 ; (i<anfang+12) && (i<anza
            hl) ; i++)
[N28]  528:     outtextxy(x1+1Ø,(i-anfang+1)*12+y1+35,f
            ile[i].name);
[U7Ø]  529: }
[ØØØ]  53Ø:
[4U6]  531: /*.....................................
              ..............*/
[ØØØ]  532:
[AW6]  533: int get_directory(char *such_parameter)
[ØØØ]  534:
[5Z8]  535: /* Liest das Inhaltsverzeichnis nach such_p
            arameter,
[LA7]  536:    Schreibt die Dateiliste in die Variable
            file,
[F46]  537:    Liefert Anzahl der Dateien zurück */
[P7Ø]  538: {
[4S5]  539:   int i,                  /* Schleifenzähler
            */
[BG7]  54Ø:       weiter,             /* noch weitere Da
            teien zu lesen */
[316]  541:       dateien,           /* Anzahl der Datei
            en */
[SR7]  542:       verzeichnisse;     /* Anzahl der Verze
            ichnisse */
[CV9]  543:   struct ffblk datei;    /* Parameter für V
            erzeichnissuche */
[5BA]  544:   int  strcmp();          /* Funktion, welc
            he die Dateien sortiert */
[ØØØ]  545:
[C32]  546:   mouse_uhr();
[ET5]  547:   for(i=Ø ; i<MAX_VERZEICHNISSE ; i++)
[4W3]  548:     strcpy(file[i].name,"");
[VZ2]  549:   verzeichnisse = Ø;
[RA5]  55Ø:   weiter = findfirst("*.*",&datei,16);
[NY2]  551:   while(weiter != -1)
[P7Ø]  552:   {
[XU3]  553:     if (datei.ff_attrib==16)
[P7Ø]  554:     {
[CP6]  555:       strcpy(file[verzeichnisse].name,"\\")
            ;
[P78]  556:       strcat(file[verzeichnisse].name,datei
            .ff_name);
[4Z3]  557:     } else verzeichnisse--;

[ØKA]  558:     if (strcmp(file[verzeichnisse].name,"\\
            .")==Ø) verzeichnisse--;
[E34]  559:     weiter = findnext(&datei);
[YP2]  56Ø:     verzeichnisse++;
[U7Ø]  561:   }
[C74]  562:   dateien = verzeichnisse;
[E97]  563:   weiter = findfirst(such_parameter,&datei,
            32);
[NY2]  564:   while(weiter != -1)
[P7Ø]  565:   {
[W37]  566:     strcpy(file[dateien].name,datei.ff_name
            );
[E34]  567:     weiter = findnext(&datei);
[BN1]  568:     dateien++;
[U7Ø]  569:   }
[VG2]  57Ø:   mouse_pfeil();
```

Listing 4. »grafik.c« enthält eine Unit, die die Grafikausgabe und Dateiverwaltung übernimmt

```
[SN2]  571:   return(dateien);
[U7Ø]  572: }
[ØØØ]  573:
[4U6]  574: /*.....................................
            ................*/
[ØØØ]  575:
[5F9]  576: void error_box(char *text)    /* Text, der
            ausgegeben wird */
[ØØØ]  577:
[EHA]  578: /* Fehlermeldung ausgeben und warten bis "O
            K" angeklickt wurde. */
[P7Ø]  579: {
[S46]  58Ø:   int lang;    /* Stringlänge in Pixel */
[ØØØ]  581:
[VG2]  582:   mouse_pfeil();
[PØ2]  583:   mouse_on();
[2C4]  584:   lang = strcspn(text,"|") * 8;
[E59]  585:   box(31Ø-lang/2,1ØØ,33Ø+lang/2,18Ø,"!! Feh
            ler !!",text,"  (O)K  ","","",
[ZL2]  586:       BLACK,LIGHTRED);
[U7Ø]  587: }
[ØØØ]  588:
[4U6]  589: /*.....................................
            ................*/
[ØØØ]  59Ø:
[8XA]  591: int box(int x1,int y1,int x2,int y2,
                 /* Koordinaten der Box */
[1F8]  592:       char *titel,
                 /* Überschrift der Box */
[377]  593:       char *text,
                 /* Text in der Box */
[KØB]  594:       char *knopf1,char *knopf2,char *kno
            pf3,    /* Beschriftung der Knöpfe */
[VU7]  595:       int color,
                 /* Vordergrundfarbe */
[T38]  596:       int h_color)
                 /* Hintergrundfarbe */
[ØØØ]  597:
[7HA]  598: /* Box an den gegebenen Koordinaten mit Tit
            el und Text ausgeben;
[LTA]  599:    knopf1,knopf2 und knopf3 enthalten die B
            eschriftung der Knöpfe,
[WR7]  6ØØ:    knopfn == "" => dieser Knopf wird nicht
            benutzt.
[56B]  6Ø1:    Ein Knopf, bei dem ein Buchstaben in run
            den Klammern geschrieben ist,
[E57]  6Ø2:    kann auch mit der Tastatur gewählt werde
            n.
[RA8]  6Ø3:    Zurückgegeben wird die gewählte Knopfnum
            mer. */
[P7Ø]  6Ø4: {
[ZE7]  6Ø5:   void *picptr;             /* Bildzwischens
            peicher */
[5C7]  6Ø6:   char *umbruch;           /* eventl. Zeilen
            umbruch */
[XH5]  6Ø7:   int x,y,                 /* Mausposition *
            /
[AK5]  6Ø8:       knopf,               /* gewählter Knop
            f */
[WV5]  6Ø9:       taste,               /* Maus und Tasta
            tur */
[LZ4]  61Ø:       mitte,               /* Fenstermitte *
            /
[4N8]  611:       key[3],               /* evtl. Buchsta
            be in runden Klammern */
[G25]  612:       i,                   /* Schleifenzähle
            r */
[NL8]  613:       l_k1,l_k2,l_k3;       /* Länge der Kno
            pfbeschriftung */
[ØØØ]  614:
[U12]  615:   mouse_off();
[DT6]  616:   picptr = getimage_in_puffer(x1,y1,x2,y2);
[CM7]  617:   putbox(x1,y1,x2,y2,titel,text,color,h_col
            or);
[C24]  618:   l_k1=textwidth(knopf1);
[F24]  619:   l_k2=textwidth(knopf2);
[K24]  62Ø:   l_k3=textwidth(knopf3);
[1Y2]  621:   mitte = x1+(x2-x1)/2;
[NK9]  622:   set_knopf(mitte-l_k1/2-3,y1+52,mitte+l_k1
            /2+3,y1+64,knopf1);
[E34]  623:   key[Ø] = get_key(knopf1);
[X43]  624:   key[1] = key[2] = Ø;
[9L2]  625:   if (knopf2[Ø]!=Ø)
[P7Ø]  626:   {
[5L9]  627:     set_knopf(mitte-l_k2/2-3,y1+68,mitte+l_
            k2/2+3,y1+8Ø,knopf2);
[M34]  628:     key[1] = get_key(knopf2);
[U7Ø]  629:   }
[CL2]  63Ø:   if (knopf3[Ø]!=Ø)
[P7Ø]  631:   {
[YL9]  632:     set_knopf(mitte-l_k3/2-3,y1+84,mitte+l_
            k3/2+3,y1+96,knopf3);
[U34]  633:     key[2] = get_key(knopf3);
[U7Ø]  634:   }
[PØ2]  635:   mouse_on();
[XG1]  636:   knopf = Ø;
[NBØ]  637:   do
[P7Ø]  638:   {
[NBØ]  639:     do
[P7Ø]  64Ø:     {
[V24]  641:       mouse_info(&x,&y,&taste);
[YN1]  642:       if (kbhit())
[P7Ø]  643:       {
[XV4]  644:         taste = toupper(get_taste());
[2S2]  645:         for(i=Ø ; i<3 ; i++)
[L15]  646:           if (taste == key[i]) knopf = i+1;
[UG1]  647:         taste = Ø;
[U7Ø]  648:       }
[Z15]  649:     } while((taste==Ø) && (knopf==Ø));
[NU9]  65Ø:     if ((x>mitte-l_k1/2-3) && (x<mitte+l_k1
            /2+3) && (y>y1+52) && (y<y1+64) &&
[Ø61]  651:         !knopf)
[ZG1]  652:       knopf = 1;
[ZU9]  653:     if ((x>mitte-l_k2/2-3) && (x<mitte+l_k2
            /2+3) && (y>y1+68) && (y<y1+8Ø) &&
[2F3]  654:         (knopf2[Ø]!=Ø) && !knopf)
[1H1]  655:       knopf = 2;
[PV9]  656:     if ((x>mitte-l_k3/2-3) && (x<mitte+l_k3
            /2+3) && (y>y1+84) && (y<y1+96) &&
[2F3]  657:         (knopf2[Ø]!=Ø) && !knopf)
[3H1]  658:       knopf = 3;
[NBØ]  659:     do
[P7Ø]  66Ø:     {
[V24]  661:       mouse_info(&x,&y,&taste);
[FY2]  662:     } while(taste!=Ø);
[A63]  663:   } while (knopf==Ø);
[U12]  664:   mouse_off();
[HF5]  665:   putimage(x1,y1,picptr,COPY_PUT);
[ØR2]  666:   farfree(picptr);
[PØ2]  667:   mouse_on();
[3D2]  668:   return(knopf);
[U7Ø]  669: }
[ØØØ]  67Ø:
[4U6]  671: /*.....................................
            ................*/
[ØØØ]  672:
[2Y3]  673: char get_key(char *text)
[ØØØ]  674:
[ØDB]  675: /* Sucht den String text nach dem Zeichen '
            (' ab und liefert das Zeichen
[G63]  676:    dahinter zurück */
[P7Ø]  677: {
[SR4]  678:   char *key;    /* Zeiger auf '(' */
[ØØØ]  679:
[ZM3]  68Ø:   key = strchr(text,'(');
[7G6]  681:   if (key!=NULL) return(toupper(key[1]));
[TL1]  682:   return(Ø);
[U7Ø]  683: }
[ØØØ]  684:
[4U6]  685: /*.....................................
            ................*/
[ØØØ]  686:
[92A]  687: void putbox(int x1,int y1,int x2,int y2,
              /* Koordinaten */
[158]  688:           char *titel,
              /* Überschrift der Box */
[3X6]  689:           char *text,
             /* Text in der Box */
[VG7]  69Ø:           int color,
              /* Vordergrundfarbe */
[TT7]  691:           int h_color)
              /* Hintergrundfarbe */
[ØØØ]  692:
[UGA]  693: /* Box an den gegebenen Koordinaten mit Tit
            el und Text ausgeben.
[W4C]  694:    Enthält text das '|' Zeichen, wird der T
            ext hinter diesem Zeichen in der
[DU4]  695:    nächsten Zeile ausgegeben. */
[P7Ø]  696: {
[KD5]  697:   int mitte;          /* Mitte der Box */
[N57]  698:   char *umbruch;      /* Zeiger auf '|' im
            String */
[ØØØ]  699:
[LR9]  7ØØ:   setfillstyle(SOLID_FILL,h_color);
                /* Box ausgeben */
[5K2]  7Ø1:   bar(x1,y1,x2,y2);
[CT2]  7Ø2:   setcolor(color);
[NK4]  7Ø3:   rectangle(x1+2,y1+2,x2-2,y2-2);
[1Y2]  7Ø4:   mitte = x1+(x2-x1)/2;
[CE2]  7Ø5:   if (titel[Ø]!=Ø)
[P7Ø]  7Ø6:   {
[748]  7Ø7:     outtextxy(mitte-textwidth(titel)/2,y1+5
            ,titel);
[Z34]  7Ø8:     if (getgraphmode()==EGAHI)
[P7Ø]  7Ø9:     {
[CT2]  71Ø:       setcolor(WHITE);
[BC8]  711:       outtextxy(mitte-textwidth(titel)/2-2,
            y1+6,titel);
[CT2]  712:       setcolor(color);
[U7Ø]  713:     }
[YZ3]  714:     line(x1+3,y1+15,x2-3,y1+15);
[U7Ø]  715:   }
[8D4]  716:   umbruch = strchr(text,'|');
[S23]  717:   if (umbruch != NULL)
[P7Ø]  718:   {
[9E2]  719:     umbruch[Ø] = Ø;
```

```
[AY7]  72Ø:      outtextxy(mitte-textwidth(text)/2,y1+23
               ,text);
[D4A]  721:      outtextxy(mitte-textwidth(&umbruch[1])/
               2,y1+35,&umbruch[1]);
[FU2]  722:      umbruch[Ø] = '|';
[SW8]  723:    } else outtextxy(mitte-textwidth(text)/2,
               y1+23,text);
[U7Ø]  724: }
[ØØØ]  725:
[4U6]  726: /*.....................................
               ................*/
[ØØØ]  727:
[ATB]  728: void set_knopf(int x1,int y1,int x2,int y2,
            /* Koordinaten des Knopfs */
[ØT7]  729:                 char *text)
            /* Beschriftung des Knopfs */
[ØØØ]  73Ø:
[D6A]  731: /* Gibt einen Knopf an den gegebenen Koordi
               naten mit Text aus. */
[P7Ø]  732: {
[3X5]  733:   int lang;       /* Textlänge in Pixel */
[ØØØ]  734:
[VW3]  735:   lang = textwidth(text);
[5R2]  736:   setcolor(BLACK);
[YP5]  737:   setfillstyle(SOLID_FILL,WHITE);
[5K2]  738:   bar(x1,y1,x2,y2);
[2N6]  739:   outtextxy(x1+1+(x2-x1)/2-lang/2,y1+3,text
               );
[7N3]  74Ø:   rectangle(x1,y1,x2,y2);
[U7Ø]  741: }
[ØØØ]  742:
[4U6]  743: /*.....................................
               ................*/
[ØØØ]  744:
[6CA]  745: void invertimage(int x1,int y1,int x2,int y
               2)    /* Koordinaten */
[ØØØ]  746:
[H67]  747: /* Invertiert einen Bildschirmausschnitt */
[P7Ø]  748: {
[XV6]  749:   void *picptr;       /* Bildzwischenspeiche
               r */
[ØØØ]  75Ø:
[DT6]  751:   picptr = getimage_in_puffer(x1,y1,x2,y2);
[AC5]  752:   putimage(x1,y1,picptr,NOT_PUT);
[ØR2]  753:   farfree(picptr);
[U7Ø]  754: }
[ØØØ]  755:
[4U6]  756: /*.....................................
               ................*/
[ØØØ]  757:
[T3C]  758: void far *getimage_in_puffer(int x1,int y1,
               int x2,int y2)  /* Koordinaten */
[ØØØ]  759:
[ZR9]  76Ø: /*  Speicher den Bildbereich in eine dynami
               sche Variable ab.
[DB6]  761:     Liefert die Speicheradresse zurück.
[CØ8]  762:     Speicherplatz mit farfree(..) wieder fr
               eigeben. */
[P7Ø]  763: {
[1X7]  764:   unsigned long platz;  /* Speicherplatzbed
               arf */
[FB5]  765:   int x,y,              /* Mausposition */
[474]  766:       taste;            /* Maustaste */
[TE8]  767:   void far *picptr;      /* Zeiger auf Spei
               cherbereich */
[ØØØ]  768:
[9Y4]  769:   platz = imagesize(x1,y1,x2,y2);
[K64]  77Ø:   picptr = farmalloc(platz);
[ØS6]  771:   if ((picptr == NULL) || (platz == ØxFFFF)
               )
[P7Ø]  772:   {
[L66]  773:     putbox(15Ø,1ØØ,49Ø,18Ø,"!! Fataler Fehl
               er !!",
[AM9]  774:            "Speicherplatz zu gering !|Progra
               mm beendet.",BLACK,RED);
[S96]  775:     set_knopf(3ØØ,15Ø,34Ø,152,"ENDE [Taste]
               ");
[VG2]  776:     mouse_pfeil();
[PØ2]  777:     mouse_on();
[NBØ]  778:     do
[P7Ø]  779:     {
[NBØ]  78Ø:       do
[P7Ø]  781:       {
[V24]  782:         mouse_info(&x,&y,&taste);
[CN2]  783:       } while(taste);
[V24]  784:       mouse_info(&x,&y,&taste);
[6R7]  785:       if ((x>3ØØ) && (x<34Ø) && (y>15Ø) &&
               (y<152) && taste) ende();
[YK3]  786:       if (get_taste()) ende();
[TV1]  787:     } while(1);
[U7Ø]  788:   }
[TR4]  789:   getimage(x1,y1,x2,y2,picptr);
[GM2]  79Ø:   return(picptr);
[U7Ø]  791: }
```

Listing 4. »grafik.c« enthält eine Unit, die die Grafikausgabe und Dateiverwaltung übernimmt

```
[YLØ]    1: ZE.C
[LL1]    2: EGAVGA.OBJ
[5C1]    3: HERC.OBJ
[KD1]    4: LITT.OBJ
```

Listing 5. Die Projektdatei »ze.prj« erleichtert das Erzeugen des ausführbaren Zeichensatzeditors »ze.exe«.

```
[165]    1: 5Ø 49 43 41 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø ØØ 1A ØØ 1D
[8T5]    2: 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D
[8T5]    3: 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D
[8T5]    4: 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D
[8T5]    5: 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D
[8T5]    6: 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D
[HR5]    7: 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D Ø7
[ØZ4]    8: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]    9: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   1Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   11: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   12: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   13: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   14: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[955]   15: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ Ø2 AØ ØØ Ø5 54 ØØ ØA AA
[475]   16: 2Ø 15 55 5Ø ØA AA 2Ø Ø5 54 ØØ Ø2 AØ ØØ ØØ ØØ ØØ
[ØZ4]   17: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   18: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[B15]   19: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ 15 ØØ ØØ ØA 8Ø ØØ
[HZ4]   2Ø: 15 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[W15]   21: ØØ ØØ 15 ØØ ØØ ØA 8Ø ØØ 15 ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   22: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   23: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[Y05]   24: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ 44 ØØ ØØ 82 ØØ ØØ 44
[S95]   25: ØØ ØØ 82 ØØ 15 55 5Ø ØA AA AØ 15 55 5Ø ØØ 82 ØØ
[C25]   26: ØØ 44 ØØ ØØ 82 ØØ ØØ 44 ØØ ØØ 82 ØØ ØØ 44 ØØ ØØ
[P85]   27: 82 ØØ ØØ 44 ØØ ØØ 82 ØØ ØØ 44 ØØ ØA AA AØ 15 55
[K65]   28: 5Ø ØA AA AØ ØØ 44 ØØ ØØ 82 ØØ ØØ 44 ØØ ØØ ØØ ØØ
[MZ4]   29: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ Ø5
[V35]   3Ø: 4Ø ØØ Ø2 AØ ØØ Ø4 1Ø 4Ø Ø8 2Ø 2Ø Ø4 1Ø 4Ø Ø8 2Ø
[X85]   31: 2Ø Ø4 1Ø 4Ø Ø8 2Ø 2Ø Ø4 1Ø 4Ø 1B EF BØ Ø4 1Ø 4Ø
[L85]   32: 1B EF BØ Ø4 1Ø 4Ø Ø8 2Ø 2Ø Ø4 1Ø 4Ø Ø8 2Ø 2Ø Ø4
[F25]   33: 1Ø 4Ø Ø8 2Ø 2Ø ØØ 15 4Ø ØØ ØA 8Ø ØØ ØØ ØØ ØØ ØØ
[ØZ4]   34: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   35: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[745]   36: ØØ ØØ ØØ ØØ 1Ø ØF ØØ 2Ø 1Ø 8Ø 5Ø Ø9 ØØ AØ 1Ø 81
[P65]   37: 4Ø Ø9 Ø2 8Ø 1Ø 85 ØØ ØF ØA ØØ ØØ 14 ØØ ØØ 28 ØØ
[W55]   38: ØØ 5Ø ØØ ØØ A1 EØ Ø1 42 1Ø Ø2 81 2Ø Ø5 Ø2 1Ø ØA
[925]   39: Ø1 2Ø 14 Ø2 1Ø Ø8 Ø1 EØ 1Ø ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   4Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[S95]   41: Ø2 82 8Ø Ø1 45 4Ø Ø6 AA 8Ø Ø8 54 4Ø Ø4 28 2Ø Ø8
[X75]   42: 1Ø 4Ø Ø4 2Ø 2Ø Ø8 1Ø 4Ø Ø4 2Ø 2Ø 1B DF DØ Ø4 2Ø
[E35]   43: 2Ø Ø8 1Ø 4Ø Ø4 2Ø 2Ø Ø8 1Ø 4Ø Ø4 2Ø 2Ø Ø8 1Ø 4Ø
[SØ5]   44: Ø4 ØØ 2Ø Ø8 ØØ 4Ø ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   45: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   46: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   47: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[LØ5]   48: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ 14 ØØ ØØ Ø8 8Ø
[H15]   49: ØØ 15 ØØ ØØ ØA ØØ ØØ 14 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   5Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   51: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[4Ø5]   52: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ 78 ØØ
[N75]   53: Ø1 86 ØØ Ø6 79 8Ø Ø9 86 4Ø 16 Ø1 AØ 28 ØØ 5Ø 1Ø
[CZ4]   54: ØØ 2Ø 2Ø ØØ 1Ø ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   55: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   56: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   57: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ

[825]   58: ØØ ØØ ØØ ØØ 2Ø ØØ 1Ø 1Ø ØØ 2Ø 28 ØØ 5Ø 16 Ø1 AØ
[X55]   59: Ø9 86 4Ø Ø6 79 8Ø Ø1 86 ØØ ØØ 78 ØØ ØØ ØØ ØØ ØØ
[ØZ4]   6Ø: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   61: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[CZ4]   62: ØØ ØØ ØØ ØØ ØØ ØØ 2Ø ØØ ØØ 1Ø ØØ ØØ 2Ø ØØ ØØ ØØ
[H85]   63: 1Ø ØØ Ø8 2Ø 8Ø 14 11 4Ø ØA 22 8Ø Ø5 15 ØØ Ø2 AA
[Z15]   64: ØØ Ø1 54 ØØ ØØ A8 ØØ ØØ 5Ø ØØ ØØ 2Ø ØØ ØØ 5Ø ØØ
[L85]   65: ØØ A8 ØØ Ø1 54 ØØ Ø2 AA ØØ Ø5 15 ØØ ØA 22 8Ø 14
[FØ5]   66: 11 4Ø Ø8 2Ø 8Ø ØØ 1Ø ØØ ØØ 2Ø ØØ ØØ 1Ø ØØ ØØ ØØ
[ØZ4]   67: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[GZ4]   68: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ Ø8 ØØ ØØ ØØ
[S15]   69: 1Ø ØØ ØØ Ø8 ØØ ØØ 1Ø ØØ ØØ Ø8 ØØ ØØ 1Ø ØØ ØØ Ø8
[Z45]   7Ø: ØØ Ø1 55 4Ø ØØ AA 8Ø Ø1 55 4Ø ØØ Ø8 ØØ ØØ 1Ø ØØ
[615]   71: ØØ Ø8 ØØ ØØ 1Ø ØØ ØØ Ø8 ØØ ØØ 1Ø ØØ ØØ Ø8 ØØ ØØ
[ØZ4]   72: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   73: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   74: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[6Z4]   75: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ 22 ØØ ØØ
[K45]   76: 5Ø ØØ ØØ A6 ØØ ØØ 58 ØØ ØØ A4 ØØ ØØ 58 ØØ ØØ 2Ø
[ØZ4]   77: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   78: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[U05]   79: ØØ ØØ 1Ø ØØ ØØ Ø8 ØØ ØØ 1Ø ØØ ØØ Ø8 ØØ ØØ 1Ø ØØ
[615]   8Ø: ØØ Ø8 ØØ ØØ 1Ø ØØ ØØ Ø8 ØØ ØØ 1Ø ØØ ØØ Ø8 ØØ ØØ
[S15]   81: 1Ø ØØ ØØ Ø8 ØØ ØØ 1Ø ØØ ØØ Ø8 ØØ ØØ 1Ø ØØ ØØ Ø8
[P05]   82: ØØ ØØ 1Ø ØØ ØØ Ø8 ØØ ØØ 1Ø ØØ ØØ Ø8 ØØ ØØ ØØ ØØ
[ØZ4]   83: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   84: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   85: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[UZ4]   86: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ 2Ø ØØ ØØ 5Ø ØØ
[8Ø5]   87: ØØ 2Ø ØØ ØØ 5Ø ØØ ØØ 2Ø ØØ ØØ 5Ø ØØ ØØ 2Ø ØØ ØØ
[ØZ4]   88: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[ØZ4]   89: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[615]   9Ø: ØØ ØØ ØØ ØØ Ø4 ØØ ØØ Ø8 ØØ ØØ 14 ØØ ØØ 28 ØØ ØØ
[235]   91: 5Ø ØØ ØØ AØ ØØ Ø1 4Ø ØØ Ø2 8Ø ØØ Ø5 ØØ ØØ ØA ØØ
[735]   92: ØØ 14 ØØ ØØ 28 ØØ ØØ 5Ø ØØ ØØ AØ ØØ Ø1 4Ø ØØ Ø2
[Z25]   93: 8Ø ØØ Ø5 ØØ ØØ ØA ØØ ØØ 14 ØØ ØØ 28 ØØ ØØ 5Ø ØØ
[AZ4]   94: ØØ 2Ø ØØ ØØ 4Ø ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
[W85]   95: ØØ FØ ØØ Ø7 ØE ØØ Ø8 F1 8Ø 17 ØE 4Ø Ø8 Ø1 8Ø 1Ø
[SØ5]   96: ØØ 4Ø 2Ø ØØ 2Ø 1Ø ØØ 4Ø 2Ø ØØ 2Ø 1Ø ØØ 4Ø 2Ø ØØ
[XØ5]   97: 2Ø 1Ø ØØ 4Ø 2Ø ØØ 2Ø 1Ø ØØ 4Ø 2Ø ØØ 2Ø 1Ø ØØ 4Ø
```

Listing 6. »caps.fnt« ist ein Pica-Beispielfont

```
[X45]    98: 20 00 20 10 00 40 00 00 00 18 01 C0 07 0E 00 08
[A65]    99: F1 80 07 0E 00 00 F0 00 00 00 00 00 00 00 00 00
[0Z4]   100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[GZ4]   101: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04
[475]   102: 00 00 00 00 00 0C 00 00 00 00 00 1F FF E0 00 00
[W55]   103: 00 3F FF E0 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]   104: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[UZ4]   105: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 06 05
[465]   106: 40 08 0A A0 16 15 40 08 08 20 10 10 40 20 20 20
[R15]   107: 10 10 40 20 20 20 10 10 40 20 20 20 10 10 40 20
[B15]   108: 20 20 10 10 40 20 20 20 10 10 40 20 20 20 10 10
[G75]   109: 40 20 20 20 10 10 40 0C 20 20 12 10 40 0D E0 20
[215]   110: 02 00 40 01 C0 20 00 00 00 00 00 00 00 00 00 00
[925]   111: 00 00 00 00 00 10 03 80 08 00 40 10 03 A0 08 00
[325]   112: 40 10 00 20 08 00 40 10 00 20 08 00 40 10 40 20
[H35]   113: 08 20 40 10 40 20 08 20 40 10 40 20 08 20 40 10
[A45]   114: 40 20 08 20 40 10 40 20 08 30 40 10 48 20 08 90

[M75]   115: 40 11 48 A0 0E 97 40 11 08 80 0E 07 00 00 00 00
[2Z4]   116: 00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 00
[R35]   117: 50 00 00 A0 00 01 50 00 02 A0 00 05 10 00 0A 20
[P05]   118: 00 14 10 00 08 20 00 10 10 00 00 20 00 00 10 00
[B25]   119: 00 20 00 00 10 00 00 20 00 00 10 00 00 20 00 1F
[3A5]   120: DF E0 00 20 00 3F DF E0 00 20 00 00 10 00 00 20
[0Z4]   121: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[V75]   122: 00 00 00 1F 80 20 20 40 40 1F A0 20 20 40 40 10
[625]   123: 20 20 20 40 40 10 20 20 20 40 40 10 20 20 20 40
[325]   124: 40 10 20 20 20 40 40 10 20 20 20 40 40 10 20 20
[725]   125: 20 40 40 10 20 20 20 40 40 10 20 20 20 40 40 10
[C35]   126: 39 A0 20 06 40 10 19 80 00 06 00 00 00 00 00 00
[225]   127: 00 00 00 00 00 00 00 00 00 00 00 01 F0 00 06 0C 00
[R65]   128: 09 F3 00 06 0C 80 18 13 40 00 20 80 10 40 60 20
[225]   129: 20 00 10 40 60 20 20 00 10 40 60 20 20 00 10 40
[425]   130: 60 20 20 00 10 40 60 20 20 00 10 40 60 20 20 00
[B55]   131: 10 40 60 08 20 80 14 53 40 08 2C 80 04 13 00 00
[T05]   132: 0C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[GZ4]   133: 00 20 00 00 10 00 00 20 00 00 10 00 00 20 00 00
[PZ4]   134: 10 00 00 20 00 00 10 00 00 20 00 00 10 00 00 20
[E65]   135: 0F E0 10 10 00 20 2F E0 10 10 00 20 60 00 10 80
[W15]   136: 00 21 40 00 10 80 00 23 00 00 14 00 00 2A 00 00
[S05]   137: 14 00 00 28 00 00 10 00 00 00 00 00 00 00 00 00
[0Z4]   138: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[N85]   139: 00 07 0F 00 08 90 80 07 0F 40 18 B0 80 00 40 40
[W15]   140: 10 20 20 20 40 40 10 20 20 20 40 40 10 20 20 20
[K35]   141: 40 40 10 20 20 20 40 40 10 20 20 00 40 40 18 B0
[F65]   142: 80 07 4F 40 08 90 80 07 0F 00 00 00 00 00 00 00
[4Z4]   143: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01
[K55]   144: C0 00 06 20 00 09 C0 00 16 20 00 08 10 00 10 20
[205]   145: 00 20 10 00 10 20 00 20 10 00 10 20 00 20 10 00
[W05]   146: 10 20 00 20 10 00 10 20 00 20 10 20 10 20 40 20
[M75]   147: 10 A0 10 21 40 00 12 80 18 25 00 04 0A 00 0B F4
[F05]   148: 00 04 08 00 03 F0 00 00 00 00 00 00 00 00 00 00
[0Z4]   149: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]   150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]   151: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[Z25]   152: 00 14 50 00 08 20 00 14 50 00 08 20 00 14 50 00
[0Z4]   153: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]   154: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]   155: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]   156: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[245]   157: 00 00 00 00 00 00 00 00 00 14 52 00 08 24 00 14 5A
[Z05]   158: 00 08 24 00 14 58 00 00 00 00 00 00 00 00 00 00
[0Z4]   159: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[T25]   160: 00 00 00 00 00 20 00 00 50 00 00 A8 00 01 54 00
[175]   161: 02 8A 00 05 05 00 0A 02 80 14 01 40 28 00 A0 50
[SZ4]   162: 00 50 20 00 20 40 00 10 00 00 00 00 00 00 00 00
[0Z4]   163: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]   164: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]   165: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[205]   166: 00 00 00 00 00 00 00 00 00 00 00 22 00 00 44 00
[215]   167: 00 22 00 00 44 00 00 22 00 00 44 00 00 22 00 00
[Y15]   168: 44 00 00 22 00 00 44 00 00 22 00 00 44 00 00 22
[E05]   169: 00 00 44 00 00 22 00 00 44 00 00 00 00 00 00 00
[0Z4]   170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]   171: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

[0Z4]   172: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[Y15]   173: 00 00 00 00 00 00 00 40 00 10 20 00 20 50 00 50 28
[565]   174: 00 A0 14 01 40 0A 02 80 05 05 00 02 8A 00 01 54
[N15]   175: 00 00 A8 00 00 50 00 00 20 00 00 00 00 00 00 00
[0Z4]   176: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[V15]   177: 00 00 00 00 00 02 00 00 05 00 00 0A 00 00 04 00
[055]   178: 00 08 00 00 10 00 00 08 14 50 10 0A 20 08 34 50
[H45]   179: 10 0A 20 08 34 50 10 00 00 08 60 00 10 00 00 08
[425]   180: 40 00 00 80 00 0B 00 00 04 80 00 03 00 00 00 00
[0Z4]   181: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[B65]   182: 00 00 00 00 00 00 00 00 00 02 AA 80 05 55 60 0A
[X45]   183: 00 10 14 00 08 08 00 10 10 00 08 08 2A 10 10 55
[V65]   184: 08 08 22 10 10 80 88 08 41 10 10 80 88 08 41 10
[095]   185: 10 80 88 08 41 10 10 80 88 08 41 10 14 00 88 0A
[235]   186: 00 10 05 55 00 02 AA 00 00 00 00 00 00 00 00 00
[A45]   187: 00 00 00 00 00 00 00 00 15 50 00 2A A0 00 55 50
[U45]   188: 00 A8 00 01 50 00 02 A0 00 05 10 00 0A 20 00 14
[P15]   189: 10 00 28 20 00 10 10 00 28 20 00 10 10 00 28 20
[345]   190: 00 14 10 00 0A 20 00 05 10 00 02 A0 00 01 50 00
[575]   191: 00 A0 00 00 50 00 00 2A A0 00 15 50 00 0A A0 00
[C05]   192: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 15 55
[S85]   193: 40 2A AA A0 15 55 40 28 20 20 10 40 40 20 20 20
[625]   194: 10 40 40 20 20 20 10 40 40 20 20 20 10 40 40 20
[025]   195: 20 20 10 40 40 20 20 20 10 40 40 20 20 20 10 40
[665]   196: 40 20 20 20 10 40 40 20 20 20 18 40 40 02 AA A0
[E45]   197: 0D 55 40 02 8A 80 00 00 00 00 00 00 00 00 00 00
[B55]   198: 00 00 00 00 00 03 55 00 04 AA 00 0B 55 80 04 00
[A25]   199: 00 18 00 C0 00 00 00 10 00 40 20 00 20 10 00 40
[V05]   200: 20 00 20 10 00 40 20 00 20 10 00 40 20 00 20 10
[S05]   201: 00 40 20 00 20 10 00 40 20 00 20 10 00 40 20 00
[G05]   202: 20 10 00 40 20 00 20 10 00 40 20 00 20 00 00 00
[U25]   203: 00 00 00 00 00 00 00 00 00 00 00 00 15 55 40 2A
[055]   204: AA A0 15 55 40 20 00 20 10 00 40 20 00 20 10 00
[Y05]   205: 40 20 00 20 10 00 40 20 00 20 10 00 40 20 00 20
[615]   206: 10 00 40 20 00 20 10 00 40 20 00 20 10 00 40 20
[C75]   207: 00 20 10 00 40 00 00 20 18 00 C0 07 FF 00 08 00

[C35]   208: 80 07 FF 00 00 00 00 00 00 00 00 00 00 00 00 00
[785]   209: 00 00 00 15 55 40 2A AA A0 15 55 40 20 20 20 10
[625]   210: 40 40 20 20 20 10 40 40 20 20 20 10 40 40 20 20
[D25]   211: 20 10 40 40 20 20 20 10 40 40 20 20 20 10 40 40
[X15]   212: 20 20 20 10 40 40 20 20 20 10 40 40 20 20 20 10
[905]   213: 00 40 20 00 20 10 00 40 20 00 00 20 00 00 00 00
[065]   214: 00 00 00 00 00 00 00 00 00 00 15 55 40 2A AA A0
[025]   215: 15 55 40 20 20 00 10 40 00 20 20 00 10 40 00 20
[Y05]   216: 20 00 10 40 00 20 20 00 10 40 00 20 20 00 10 40
[C05]   217: 00 20 20 00 10 40 00 20 20 00 10 40 00 20 00 00
[PZ4]   218: 10 00 00 20 00 00 10 00 00 20 00 00 10 00 00 20
[0Z4]   219: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[R55]   220: 00 01 55 00 02 AA 80 05 01 40 0A 00 A0 14 00 40
[L15]   221: 28 00 20 10 00 40 20 00 20 10 00 40 20 00 20 10
[815]   222: 00 40 20 00 20 10 20 40 20 10 20 10 20 40 20 10
[X15]   223: 20 10 20 40 20 10 20 10 20 40 20 10 20 10 28 40
[245]   224: 20 17 A0 10 08 40 00 07 80 00 00 00 00 00 00 00
[C75]   225: 00 00 00 00 00 00 00 00 15 55 40 2A AA A0 15 55
[C05]   226: 40 00 20 00 00 40 00 00 20 00 00 40 00 00 20 00
[005]   227: 00 40 00 00 20 00 00 40 00 00 20 00 00 40 00 00
[C05]   228: 20 00 00 40 00 00 20 00 00 40 00 00 20 00 00 40

[WB5]   229: 00 00 20 00 00 40 00 2A AA A0 15 55 40 2A AA A0
[0Z4]   230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]   231: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]   232: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[P75]   233: 00 00 00 15 55 40 2A AA A0 15 55 40 00 00 00 00
[0Z4]   234: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]   235: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[K15]   236: 00 00 00 00 00 00 00 07 00 00 00 80 00 07 40 00
[H05]   237: 00 A0 00 00 40 00 00 20 00 00 40 00 00 20 00 00
[M55]   238: 40 00 00 20 00 00 40 00 00 20 00 00 40 2A AA A0
[R55]   239: 15 55 40 2A AA 80 00 00 00 00 00 00 00 00 00 00
[0Z4]   240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[K05]   241: 00 00 00 00 00 00 00 00 00 00 00 00 00 35 55 40
[W65]   242: 0A AA A0 35 55 40 00 20 00 00 40 00 00 20 00 00
[E05]   243: 40 00 00 20 00 00 40 00 00 20 00 00 40 00 00 20
[935]   244: 00 00 40 00 00 20 00 00 50 00 00 A8 00 01 54 00
[N65]   245: 02 8A 00 05 05 00 0A 02 80 14 01 40 28 00 A0 10
[GZ4]   246: 00 40 20 00 20 00 00 00 00 00 00 00 00 00 00 00
[U75]   247: 00 00 00 00 15 55 40 2A AA A0 15 55 40 00 00 20
[G05]   248: 00 00 40 00 00 20 00 00 40 00 00 20 00 00 40 00
[205]   249: 00 20 00 00 40 00 00 20 00 00 40 00 00 20 00 00
[E05]   250: 40 00 00 20 00 00 40 00 00 20 00 00 40 00 00 20
[005]   251: 00 00 40 00 00 20 00 00 40 00 00 20 00 00 00 00
[W55]   252: 00 00 00 00 00 00 00 00 00 00 2A AA A0 15 55
[F85]   253: 40 2A AA A0 14 00 00 0A 00 00 05 00 00 02 80 00
[H15]   254: 01 40 00 00 A0 00 00 50 00 00 28 00 00 14 00 00
[L35]   255: 28 00 00 50 00 00 A0 00 01 40 00 02 80 00 05 00
[0D5]   256: 00 0A 00 00 14 00 00 2A AA A0 15 55 40 2A AA A0
[0Z4]   257: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[MC5]   258: 00 00 2A AA A0 15 55 40 2A AA A0 14 00 00 0A 00
[M15]   259: 00 05 00 00 02 80 00 01 40 00 00 80 00 00 60 00
[G15]   260: 00 00 00 00 30 00 00 08 00 00 14 00 00 0A 00 00
[6C5]   261: 05 00 00 00 02 80 00 01 40 2A AA A0 15 55 40 2A AA
[205]   262: A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[H05]   263: 00 00 00 00 00 00 00 00 00 00 00 00 00 7C 00 01
[695]   264: 83 00 06 7C 80 09 83 00 06 00 C0 18 00 00 00 00
[X05]   265: 40 30 00 20 00 00 40 30 00 20 00 00 40 30 00 20
[Z15]   266: 00 00 40 30 00 20 00 00 40 30 00 20 00 00 40 18
[Y85]   267: 00 00 06 00 C0 09 83 00 06 7C 80 01 83 00 00 7C
[0Z4]   268: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[D85]   269: 15 55 50 2A AA A0 15 55 50 20 20 00 10 10 00 20
[905]   270: 20 00 10 10 00 20 20 00 10 10 00 20 20 00 10 10
[805]   271: 00 20 20 00 10 10 00 20 20 00 10 10 00 20 20 00
[G15]   272: 10 10 00 20 20 00 10 10 00 20 20 00 18 40 00 05
[S15]   273: 20 00 0A C0 00 05 00 00 00 00 00 00 00 00 00 00
[0Z4]   274: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[495]   275: 00 F8 00 03 06 00 0C F9 80 13 06 40 0C 01 80 10
[S15]   276: 00 40 20 00 20 10 00 40 20 00 20 10 00 40 20 08
[W35]   277: 20 10 14 40 20 0A 20 10 05 40 20 02 A0 10 00 40
[DB5]   278: 0C 01 A0 13 06 50 0C F9 A8 03 06 10 00 F8 00 00
[C05]   279: 00 00 00 00 00 00 00 00 00 00 00 00 00 15 55
[Z65]   280: 40 2A AA A0 15 55 40 20 20 00 10 10 00 20 20 00
[705]   281: 10 10 00 20 20 00 10 10 00 20 20 00 10 10 00 20
[H15]   282: 20 00 10 10 03 20 20 00 10 10 00 20 28 00 10 14
[X55]   283: 00 20 2A 00 10 15 00 20 22 80 1C C1 40 03 20 A0
[A25]   284: 0C C0 40 03 00 20 00 00 00 00 00 00 00 00 00 00
[R25]   285: 00 00 00 00 00 00 00 00 07 00 40 18 C0 20 07 00
[V45]   286: 40 18 C0 20 20 20 40 10 40 20 20 20 40 10 40 20
[925]   287: 20 20 40 10 40 20 20 20 40 10 40 20 20 20 40 10
[B35]   288: 40 20 20 20 40 10 40 20 20 20 40 10 40 20 20 38
[D45]   289: C0 10 07 20 20 18 C0 10 07 00 00 00 00 00 00 00
[7Z4]   290: 00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 10
[SZ4]   291: 00 00 00 20 00 00 10 00 00 20 00 00 10 00 00 20 00
[B65]   292: 00 10 00 00 20 00 00 10 00 00 2F FF E0 10 00 00
[F65]   293: 2F FF E0 10 00 00 20 00 00 10 00 00 20 00 00 10
[SZ4]   294: 00 00 20 00 00 10 00 00 20 00 00 10 00 00 20 00
[1Z4]   295: 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[AB5]   296: 00 00 00 3F FC 00 00 03 00 3F FC 80 00 03 00 00
[B05]   297: 00 C0 00 00 00 00 00 40 00 00 20 00 00 40 00 00
[C05]   298: 20 00 00 40 00 00 20 00 00 40 00 00 20 00 00 40
[W05]   299: 00 00 20 00 00 40 00 00 20 00 00 40 00 00 80 00
[M95]   300: 03 40 3F FC 80 00 03 00 3F FC 00 00 00 00 00 00
[525]   301: 00 00 00 00 00 00 00 00 00 3F E0 00 00 10 00
[865]   302: 3F E8 00 00 14 00 00 0A 00 00 05 00 00 02 80 00
[115]   303: 01 40 00 00 A0 00 00 40 00 00 20 00 00 40 00 00
[S25]   304: 20 00 00 40 00 00 A0 00 01 40 00 02 80 00 05 00
[K75]   305: 00 0A 00 00 14 00 3F E8 00 00 00 10 00 3F E0 00
[0Z4]   306: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[5B5]   307: 00 3F F8 00 00 04 00 3F FA 00 00 05 00 00 02 80
[G25]   308: 00 01 40 00 00 A0 00 00 40 00 00 A0 00 01 40 00
[F15]   309: 02 80 00 05 00 00 0A 00 00 05 00 00 02 80 00 01
[235]   310: 40 00 00 A0 00 00 40 00 00 A0 00 01 40 00 02 80
[W85]   311: 3F FD 00 00 02 00 3F FC 00 00 00 00 00 00 00 00
[305]   312: 00 00 00 00 00 00 00 20 00 20 10 00 40 28 00
[A85]   313: A0 14 01 40 0A 02 80 05 05 00 02 8A 00 01 54 00
[W55]   314: 00 A8 00 00 50 00 00 A8 00 01 54 00 02 8A 00 05
[X45]   315: 05 00 0A 02 80 14 01 40 28 00 A0 10 00 40 20 00
[4Z4]   316: 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[6Z4]   317: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 20
[D25]   318: 00 00 10 00 00 28 00 00 14 00 00 0A 00 00 05 00
```

```
[665]  319: 00 02 80 00 01 40 00 00 A0 00 00 5F E0 00 20 00
[B85]  320: 00 5F E0 00 A0 00 01 40 00 02 80 00 05 00 00 0A
[205]  321: 00 00 14 00 00 28 00 00 10 00 00 20 00 00 00 00
[0Z4]  322: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[425]  323: 00 00 00 00 00 00 20 00 20 10 00 40 20 00 A0 10
[W45]  324: 01 40 20 02 A0 10 05 40 20 0A 20 10 14 40 20 28
[K35]  325: 20 10 50 40 20 A0 20 11 40 40 22 80 20 15 00 40
[335]  326: 2A 00 20 14 00 40 28 00 20 10 00 40 20 00 20 00
[4Z4]  327: 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[L05]  328: 00 00 00 00 00 00 00 00 00 00 00 00 00 15 55 50
[K75]  329: 0A AA A0 15 55 50 08 00 20 10 00 10 08 00 20 10
[N05]  330: 00 10 08 00 20 10 00 10 08 00 20 10 00 10 00 00
[0Z4]  331: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  332: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  333: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[C15]  334: 00 00 00 00 10 00 00 08 00 00 14 00 00 0A 00 00
[X25]  335: 05 00 00 02 80 00 01 40 00 00 A0 00 00 50 00 00
[W15]  336: 28 00 00 14 00 00 0A 00 00 05 00 00 02 80 00 01
[L15]  337: 40 00 00 A0 00 00 50 00 00 20 00 00 10 00 00 00
[0Z4]  338: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  339: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  340: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  341: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[815]  342: 00 00 10 00 10 08 00 20 10 00 10 08 00 20 10 00
[085]  343: 10 08 00 20 10 00 10 08 00 20 15 55 50 0A AA A0
[M05]  344: 15 55 50 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  345: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[U05]  346: 00 01 80 00 00 00 00 03 00 00 00 00 00 06 00 00
[945]  347: 00 00 00 0C 00 00 00 00 00 18 00 00 00 00 00 0C
[SZ4]  348: 00 00 00 00 00 00 06 00 00 00 00 00 03 00 00 00
[205]  349: 00 01 80 00 00 00 00 00 00 00 00 00 00 00 00 00
[6Z4]  350: 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 02
[GZ4]  351: 00 01 00 00 02 00 00 01 00 00 02 00 00 01 00 00
[SZ4]  352: 02 00 00 01 00 00 02 00 00 01 00 00 02 00 00 01
[GZ4]  353: 00 00 02 00 00 01 00 00 02 00 00 01 00 00 02 00
[G24]  354: 00 01 00 00 02 00 00 01 00 00 02 00 00 01 00 00
[SZ4]  355: 02 00 00 01 00 00 02 00 00 01 00 00 02 00 00 01
[0Z4]  356: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[T15]  357: 00 00 00 00 00 00 00 04 00 00 0A 00 00 15 00 00
[N15]  358: 00 0A 80 00 05 40 00 00 00 00 00 00 00 00 00 00
[0Z4]  359: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  360: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[B05]  361: 00 00 00 00 00 00 00 00 00 00 00 00 E0 00 01 00
[P65]  362: 00 02 E0 00 05 00 00 0A 80 00 15 00 00 28 80 00
[935]  363: 51 00 00 20 80 00 41 00 00 20 80 00 51 00 00 28
[X35]  364: 80 00 15 00 00 0A 80 00 05 00 00 02 E0 00 01 00
[M15]  365: 00 00 E0 00 00 00 00 00 00 00 00 00 00 00 00 00
[305]  366: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55
[L45]  367: 40 00 2A A0 00 55 40 00 22 20 00 44 40 00 22 20
[S25]  368: 00 44 40 00 22 20 00 44 40 00 22 20 00 44 40 00
[P45]  369: 22 20 00 44 40 00 3B A0 00 00 40 00 19 80 00 00
[0Z4]  370: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  371: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[W25]  372: 00 00 00 00 00 00 15 40 00 2A 80 00 55 40 00 20
[W15]  373: 20 00 40 40 00 20 20 00 40 40 00 20 20 00 40 40
[615]  374: 00 20 20 00 40 40 00 20 20 00 40 40 00 20 20 00
[K05]  375: 40 40 00 20 20 00 40 40 00 00 00 00 00 00 00 00
[0Z4]  376: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[ZZ4]  377: 00 00 00 00 00 00 00 00 00 00 00 00 00 55 40 00
[745]  378: 2A A0 00 55 40 00 20 20 00 40 40 00 20 20 00 40
[K15]  379: 40 00 20 20 00 40 40 00 20 20 00 40 40 00 20 20
[545]  380: 00 50 40 00 2A A0 00 15 40 00 0A 80 00 00 00 00
[0Z4]  381: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  382: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[W45]  383: 00 00 00 00 55 40 00 2A A0 00 55 40 00 22 20 00
[Y25]  384: 44 40 00 22 20 00 44 40 00 22 20 00 44 40 00 22
[S25]  385: 20 00 44 40 00 22 20 00 44 40 00 22 20 00 40 40
[PZ4]  386: 00 20 20 00 40 40 00 00 00 00 00 00 00 00 00 00
[0Z4]  387: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[R25]  388: 00 00 00 00 00 00 00 00 00 00 00 00 55 40 00 2A
[V15]  389: 00 55 40 00 22 00 00 44 00 00 22 00 00 44 00 00
[225]  390: 22 00 00 44 00 00 22 00 00 44 00 00 22 00 00 44
[WZ4]  391: 00 00 20 00 00 40 00 00 20 00 00 40 00 00 00 00
[0Z4]  392: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  393: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[D45]  394: 00 00 0F 80 00 10 00 00 0F C0 00 30 20 00 40 40
[E15]  395: 00 20 20 00 40 40 00 20 20 00 40 40 00 24 20 00
[435]  396: 42 40 00 24 20 00 42 40 00 24 20 00 42 40 00 24
[635]  397: 20 00 42 40 00 01 A0 00 02 40 00 01 80 00 00 00
[0Z4]  398: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[A35]  399: 00 00 00 00 00 00 00 00 00 55 40 00 2A A0 00 55
[A05]  400: 40 00 02 00 00 04 00 00 02 00 00 04 00 00 02 00
[805]  401: 00 04 00 00 02 00 00 04 00 00 02 00 00 04 00 00
[W35]  402: 02 00 00 55 40 00 2A A0 00 55 40 00 00 00 00 00
[0Z4]  403: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  404: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  405: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[505]  406: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 40
[535]  407: 00 2A A0 00 55 40 00 00 00 00 00 00 00 00 00 00
[0Z4]  408: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  409: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[Y05]  410: 00 00 00 00 00 00 00 01 40 00 00 80 00 01 60 00
[405]  411: 00 00 00 00 60 00 00 00 00 00 60 00 00 00 00 00
[T55]  412: 60 00 00 00 00 55 E0 00 2A 00 00 55 80 00 00 00
[0Z4]  413: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  414: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[505]  415: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 40
[K35]  416: 00 2A A0 00 55 40 00 02 00 00 01 00 00 02 00 00
[R45]  417: 05 00 00 0A 00 00 15 80 00 28 40 00 50 A0 00 20
[YZ4]  418: 40 00 40 20 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  419: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  420: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[S35]  421: 00 00 00 00 00 55 40 00 2A A0 00 55 40 00 00 20
[G05]  422: 00 00 40 00 00 20 00 00 40 00 00 20 00 00 40 00
[205]  423: 00 20 00 00 40 00 00 20 00 00 40 00 00 20 00 00
[EZ4]  424: 40 00 00 20 00 00 00 00 00 00 00 00 00 00 00 00
```

```
[0Z4]  425: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[725]  426: 00 00 00 00 00 00 00 00 00 00 00 00 00 55 40 00 2A
[G45]  427: A0 00 55 40 00 28 00 00 14 00 00 0A 00 00 05 00
[625]  428: 00 02 80 00 01 00 00 02 80 00 05 00 00 0A 00 00
[Y55]  429: 14 00 00 28 00 00 55 40 00 2A A0 00 55 40 00 00
[0Z4]  430: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  431: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[555]  432: 00 00 00 55 40 00 2A A0 00 55 40 00 28 00 00 14
[825]  433: 00 00 0A 00 00 05 00 00 02 80 00 01 40 00 00 A0
[K35]  434: 00 55 40 00 2A A0 00 55 40 00 00 00 00 00 00 00
[0Z4]  435: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  436: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[T15]  437: 00 00 00 00 00 00 00 00 00 00 15 40 00 2A 80 00
[R15]  438: 15 40 00 20 20 00 40 40 00 20 20 00 40 40 00 20
[W15]  439: 20 00 40 40 00 20 20 00 40 40 00 20 20 00 40 40
[E35]  440: 00 20 20 00 15 40 00 2A 80 00 15 40 00 00 00 00
[0Z4]  441: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  442: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[845]  443: 00 55 40 00 2A A0 00 55 40 00 22 00 00 44 00 00
[225]  444: 22 00 00 44 00 00 22 00 00 44 00 00 22 00 00 44
[335]  445: 00 00 22 00 00 44 00 00 22 00 00 5C 00 00 20 00
[705]  446: 00 1C 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  447: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[S25]  448: 00 00 00 00 00 00 00 00 05 00 00 0A 80 00 15 40
[W15]  449: 00 28 80 00 10 40 00 20 20 00 40 00 00 20 20 00
[E15]  450: 40 00 00 20 20 00 40 00 00 20 20 00 44 00 00 22
[155]  451: 20 00 41 00 00 20 A0 00 10 40 00 28 A0 00 15 50
[H15]  452: 00 0A 80 00 05 00 00 00 00 00 00 00 00 00 00 00
[305]  453: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55
[R35]  454: 40 00 2A A0 00 55 40 00 22 00 00 41 00 00 22 00
[G05]  455: 00 41 00 00 22 00 00 41 00 00 22 00 00 41 00 00
[365]  456: 22 00 00 41 00 00 22 80 00 5D 40 00 22 A0 00 1C
[EZ4]  457: 40 00 00 20 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  458: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[435]  459: 00 00 00 00 00 00 3C 60 00 00 00 00 7E 60 00 00
[435]  460: 00 00 66 60 00 00 00 00 66 60 00 00 00 00 66 60
[M15]  461: 00 00 00 00 66 60 00 00 00 00 66 60 00 00 00 00
[K25]  462: 67 E0 00 00 00 00 03 C0 00 00 00 00 00 00 00 00
[0Z4]  463: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[4Z4]  464: 00 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00
[315]  465: 20 00 00 40 00 00 20 00 00 40 00 00 20 00 00 55
[B35]  466: 40 00 2A A0 00 55 40 00 20 00 00 40 00 00 20 00
[MZ4]  467: 00 40 00 00 20 00 00 40 00 00 00 00 00 00 00 00
[0Z4]  468: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  469: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[045]  470: 00 00 00 00 55 00 00 2A 80 00 55 40 00 00 20 00
[005]  471: 00 40 00 00 20 00 00 40 00 00 20 00 00 40 00 00
[845]  472: 20 00 00 40 00 00 20 00 55 40 00 2A 80 00 55 00
[0Z4]  473: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  474: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[S05]  475: 00 00 00 00 00 00 00 00 00 00 00 54 00 00 2A 00
[C25]  476: 00 55 00 00 02 80 00 01 40 00 00 A0 00 00 40 00
[W35]  477: 00 A0 00 01 40 00 02 80 00 55 00 00 2A 00 00 54
[0Z4]  478: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  479: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  480: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[845]  481: 00 00 55 00 00 2A 80 00 55 40 00 00 A0 00 00 40
[C35]  482: 00 00 A0 00 01 40 00 02 80 00 05 00 00 02 80 00
[065]  483: 01 40 00 00 A0 00 00 40 00 00 A0 00 55 40 00 2A
[905]  484: 80 00 55 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  485: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[005]  486: 00 00 00 00 00 00 00 00 00 40 20 00 20 40 00 40
[145]  487: 20 00 30 C0 00 09 00 00 16 80 00 09 00 00 06 00
[345]  488: 00 09 00 00 16 80 00 09 00 00 30 C0 00 40 20 00
[SZ4]  489: 20 40 00 40 20 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  490: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  491: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[635]  492: 40 00 00 20 00 00 50 00 00 28 00 00 14 00 00 0A
[S45]  493: 00 00 05 40 00 02 A0 00 05 40 00 0A 00 00 14 00
[K05]  494: 00 28 00 00 50 00 00 20 00 00 40 00 00 00 00 00
[0Z4]  495: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  496: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[M05]  497: 00 00 00 00 00 00 00 00 60 20 00 00 00 00 60 60 00
[455]  498: 00 80 00 61 60 00 02 80 00 65 60 00 0A 00 00 74
[B15]  499: 60 00 08 00 00 70 60 00 00 00 00 40 60 00 00 00
[0Z4]  500: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  501: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[C15]  502: 00 00 00 00 00 00 00 00 00 00 00 00 00 07 11 C0
[T95]  503: 08 AA 20 07 55 C0 08 AA 20 10 44 10 08 00 20 10
[X05]  504: 00 10 08 00 20 10 00 10 08 00 20 10 00 10 08 00
[8Z4]  505: 20 10 00 10 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  506: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  507: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  508: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  509: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[755]  510: 00 00 00 00 00 00 00 00 15 45 50 0A 82 A0 15 45
[AZ4]  511: 50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  512: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  513: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  514: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[605]  515: 00 00 00 00 00 00 00 00 00 00 00 00 10 00 10 08
[A15]  516: 00 20 10 00 10 08 00 20 10 00 10 08 00 20 10 00
[B85]  517: 10 08 00 20 10 C6 10 09 29 20 06 D6 C0 09 29 20
[Z15]  518: 06 10 C0 00 00 00 00 00 00 00 00 00 00 00 00 00
[M05]  519: 00 00 00 00 00 00 00 04 00 00 08 00 00 14 00 00
[615]  520: 00 08 00 00 10 00 00 08 00 00 10 00 00 08 00 00
[015]  521: 04 00 00 08 00 00 02 00 00 04 00 00 02 00 00 01
[F05]  522: 00 00 02 00 00 01 00 00 02 00 00 05 00 00 0A 00
[005]  523: 00 04 00 00 08 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  524: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  525: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  526: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  527: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  528: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[0Z4]  529: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Listing 6. »caps.fnt« ist ein Pica-Beispielfont

Impressum

Herausgeber
Christian Widuch

Leitung des Redaktionsbüros
Volker Everts

Chefredakteur
Volker Everts (ev)

Stellvertretender Chefredakteur
Ulrich Eike (ue)

Projektleitung
Dieter Bäumler (db)

Redaktion
Herbert Buckel ( ), Helmut Stult ( )

Redaktionsmarketing
Ingo Everts (ie)

Schlußredaktion
Karin Adlmannseder (ka), Elisabeth Foessleitner (el)

Redaktionsassistenz
Andrea Hartl, Anke Ludwig, Christine Reichherzer

Produktionsleitung
Gerd Köberich, Uwe Siebert

Ressortleitung
Yvonne Hendricks (Layout)
Achim Schulte (Fotografie)

Abteilungsleitung
Claudia Ebbrecht (Fotosatz/Korrektur)
Margarete Schenk, Helmut Skoupy (Montage/Reprografie)

Layout
Yvonne Hendricks

Fotografie
Heinz Harbuca

Werbeleitung
Jutta Himmelmann

Werbegrafik
Mohamed Hawa, Uwe Kalischinsky

Fotosatz
Dirk Anhof

Montage/Reprografie
Manuela Eska, Andrea Gundlach

Anzeigenleitung
Wolfgang Brill

Anzeigenverkauf für PLZ 1, 4, 5
Gerlinde Rachow (0 56 51) 80 93 90
Sylvia Stephani (0 56 51) 80 93 80
Bernd Heckmann (0 56 51) 80 93 81
Karina Ehrlich (0 56 51) 80 93 71

Anzeigenverkauf für PLZ 2 + 3
DMV-Verlagsbüro Hamburg
Ohlsdorfer Straße 34, 2000 Hamburg 60
Leitung: Silvia Ehrenpfordt,
Anzeigenverkauf: Sabine Bindseil, Ralf Streng,
Tel.: (0 40) 46 12 33,
Telefax (0 40) 47 43 10

Anzeigenverkauf für PLZ 6 – 8
DMV-Verlagsbüro München
Zaunkönigweg 2c, 8000 München 82,
Telefon: (0 89) 4 39 10 87, Telefax: (0 89) 4 39 10 80
Leitung: Britta Fiebig
Anzeigenverkauf: Peter Schätzle, Hannelore David

Anzeigenverwaltung und Disposition
Andrea Giese, Karina Ehrlich, Beate Kranz, Christina Wabra

Anschrift Redaktion:
Redaktion DOS International
Gruberstraße 46c, 8011 Poing bei München
Telefon: (0 81 21) 7 69-0, Telefax: (0 81 21) 7 90 46

Anschrift Verlag:
DMV Daten- und Medien-Verlag
Widuch GmbH & Co. KG
Fuldaer Straße 6, 3440 Eschwege
Telefon: (0 56 51) 8 09-0
Telefax: (0 56 51) 80 93 33

Vertrieb
Verlagsunion Erich Pabel-Arthur Moewig KG (VPM)
Friedrich-Bergius-Straße 20
6200 Wiesbaden

Druck
Druckerei Jungfer, 3420 Herzberg

Bezugspreise
DOS Extra erscheint viermal jährlich
Einzelpreis DM 18, – , 145 öS, 18, – sfr., Fmk 55, –

Abonnementpreise
Die Preise verstehen sich grundsätzlich
einschließlich Porto und Verpackung.

Inland:
Bezugspreis 4 Ausgaben: DM 60, –

Europäisches Ausland:
Bezugspreis 4 Ausgaben: DM 70, –

Außereuropäisches Ausland:
Bezugspreis 4 Ausgaben: DM 80, –

Bankverbindungen:
Postscheck Frankfurt/M.: Kto.-Nr.: 23043-608
Raiffeisenbank Eschwege:
BLZ: 522 603 85, Kto.-Nr.: 245 7008

Die Abonnementbestellung kann innerhalb einer Woche nach Auftrag beim DMV-Verlag, Postfach 250, 3440 Eschwege, schriftlich widerrufen werden. Zur Wahrung der Frist reicht der Poststempel. Das Abonnement verlängert sich automatisch um 6 bzw. 12 Monate, wenn es nicht mindestens 6 Wochen vor Ablauf beim Verlag schriftlich gekündigt wird.

Manuskripte, Programme:
bitte an die Redaktion
DOS International, Gruberstraße 46c, 8011 Poing, senden.

Für unverlangt eingesandte Manuskripte und Datenträger sowie Fotos übernehmen der Verlag und die Redaktion keine Haftung. Die Zustimmung zum Abdruck wird vorausgesetzt. Das Urheberrecht für veröffentlichte Manuskripte liegt ausschließlich beim Verlag. Nachdruck sowie Vervielfältigung oder sonstige Verwertung von Texten nur mit schriftlicher Genehmigung des Verlages. Für veröffentlichte Programme und Basteleien übernehmen wir weder Gewähr noch Haftung. Schaltungen und verwendete Bezeichnungen müssen nicht frei von gewerblichen Schutzrechten sein. Namentlich gekennzeichnete Fremdbeiträge geben nicht in jedem Fall die Meinung der Redaktion wieder.

Mitglied der Informationsgemeinschaft zur Feststellung der Verbreitung von Werbeträgern e.V. (IVW),
Bad Godesberg.
ISSN 0933-1557

BCi DisAsm i86
Intelligenter Disassembler Version 1.0

Sesam öffne dich!

BCi DisAsm erschließt jedes Programm!

Wandelt den Programmcode in ein dokumentiertes Assemblerlisting um. Mit normalen Editoren können Sie nun verändern und modifizieren, und mit gängigen Assemblern und Debuggern kann danach wieder assembliert werden.

Es gibt kaum einen, der BCi DisAsm 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-Puzzles, welche sich beim Umgang mit dem Computer ergeben, zu lösen. Und zwar schneller als mit jedem anderen Produkt, das Sie kaufen können, egal, um welche Preisklasse es sich handelt.
Mit dem BCi DisAsm erwerben Sie sofort ein Know-how, das Sie normalerweise 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 Menüs gesteuert werden. Die Menüstruktur ist Lotus-kompatibel und selbstverständlich in Deutsch. Es besteht keine Notwendigkeit, neue Befehle zu lernen oder ständig im Handbuch nachzuschlagen. BCi DisAsm ist unmittelbar anwendbar!

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

BCi DisAsm wird Ihnen bei den folgenden Schritten helfen:

1. Lernen Sie Assembler-Programmierung, wenn Sie möchten.
2. Entdecken Sie, warum Programm XYZ auf Ihrem Rechner nicht läuft, und tun Sie 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 inkl. eines sehr ausführlichen deutschen Handbuchs (mehr als 100 Seiten) nur

199, – DM unverbindliche Preisempfehlung
Wenn Sie über den DMV-Bestellservice bestellen, gilt folgendes:

Inland:		**Ausland:**	
Einzelpreis	199, – DM	Einzelpreis	199, – DM
zzgl. Versandkosten	4, – DM	zzgl. Versandkosten	6, – DM
Endpreis	**203, – DM**	**Endpreis**	**205, – DM**

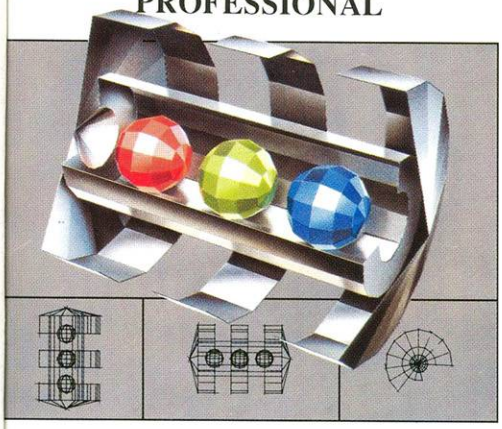
Bitte benutzen Sie die Bestellkarte.

DMV-Verlag · Postfach 250 · 3440 Eschwege

DMV Daten- und Medienverlag

DOS SOFTWARE

3D-DRAW PROFESSIONAL



Komfortables, dreidimensionales Konstruktionsprogramm

Die Leistungsmerkmale von 3D-DRAW PROFESSIONAL:

3D-Modus

• Kantenmodelle und Flächenmodelle mit und ohne Lichtquelle animierbar • anwählbare Lichtquelle • Rotation und Translation um beliebige Raumachsen • manipulierbare Bereiche • konfigurierbare Druckertreiber • 2000 Punkte und Flächen sowie 4000 Kanten darstellbar • Einzelbehandlung von Körpern • Leuchtbalken-Menüs zur besseren Anwählbarkeit

2D-Modus

• Erhöhte Arbeitsgeschwindigkeit • Mauszeiger-Bedienung auch über die Tastatur • Zurücknehmen des letzten Arbeitsschrittes • Quick-Modus für einfache Grundformen wie Kreise oder Vielecke • Werkzeuge zum Pressen, Stauchen und Dehnen von Objekten, Körpern und Bereichen

Systemvoraussetzungen: PC, XT, AT oder Kompatibler mit mindestens 640 KByte RAM. **Grafikkarte:** Alle üblichen Typen. **Ausgabemedien:** Epson-kompatible Nadeldrucker, HP- kompatible Plotter, Postscript-Drucker. **Sonstiges:** Microsoft-kompatible Maus unbedingt erforderlich, MS-DOS ab Version 3.0

nur DM 199,– *

3D-DRAW 1.0 **nur DM 99,– ***

S O F T W A R E

NAVIGATOR 4.5

Wetterberatung und Flugplanung für den Microsoft Flugsimulator III und IV

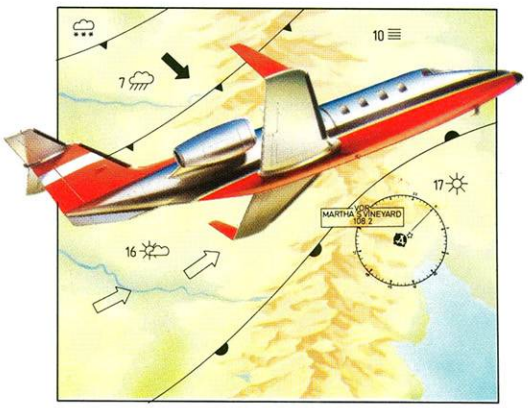
Der neu überarbeitete NAVIGATOR 4.5 bietet jetzt noch mehr Bedienungskomfort und arbeitet noch enger mit dem Microsoft Flugsimulator zusammen.

Das Leistungsspektrum auf einen Blick:

• Auf Wunsch automatische Datenübertragung an den Flugsimulator • 1237 Flughäfen stehen zur Verfügung • eine interne Datenbank bietet 610 komplette Flugrouten mit allen zu überfliegenden Funkfeuern, Kursen und Zwischenzeiten • eine weitere Datenbank enthält Informationen über 900 Funkfeuer • 400 zusätzliche Routen lassen sich auf Wunsch anfertigen • von jedem Flughafen aus sind Rundflüge möglich • NAVIGATOR 4.5 läßt sich jetzt per Maus bedienen

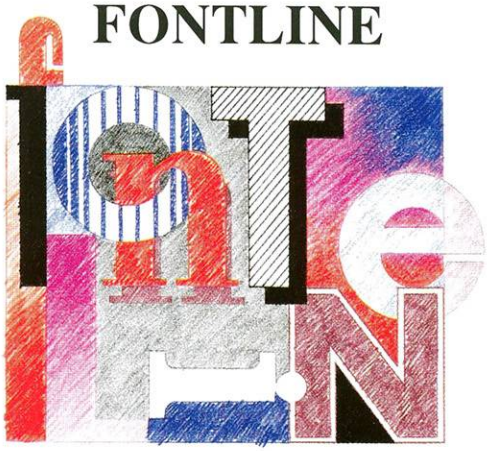
Systemvoraussetzungen:
Microsoft Flugsimulator 3.0 oder 4.0, 512 KByte Arbeitsspeicher, jede, vom Flugsimulator unterstützte Grafikkarte

nur DM 69,– *



S O F T W A R E

FONTLINE



Softfont-Manipulation und Druckertreiber-Generierung für Microsoft-Word 4.0/5.0 sowie Ventura Publisher 1.1/1.2/2.0

FONTLINE erzeugt aus bestehenden Softfonts für HP-Laserjet Plus und Serie II, sowie dazu kompatible Laserdrucker nahezu beliebig viele neue Variationen. Die bekanntesten wie Inline, Outline, 3D, Schatten, Relief, Invers, Kursiv und Landscape können sofort von allen ladbaren Zeichensätzen angefertigt werden.

Erzeugen Sie Zeichensätze mit Marmorstruktur, Chrom-Glanz, Holzmaserung oder beliebigen anderen Mustern, indem Sie bekannte Zeichenprogramme wie Publisher's Paintbrush oder Deluxe Paint II zu Hilfe nehmen.

FONTLINE macht alle vorhandenen und neu erzeugten Fonts auch für Microsoft Word 4/5 und Ventura Publisher zugänglich.

Ihr Programm unterstützt die Einbindung neuer Fonts beispielsweise in Microsoft Windows 2.1 oder Aldus Pagemaker? Dann versorgt Sie zukünftig FONTLINE mit einer umfangreichen Schriftenvielfalt.

Mitgelieferte, einsatzfertige Softfonts in diversen Größen: Alte Schwabacher, Futura Black, Bembo, Keys, Sybolic, PC-Linedraw

Systemvoraussetzungen: PC, XT, AT, PS/2 und Kompatible mit mindestens 512 KByte RAM, HP- Laserjet Plus/Serie II oder kompatibler Laserdrucker, MS-DOS 3.2 und höher

Erscheinungstermin: 1. Quartal 1991

nur DM 299,– *

S O F T W A R E

* Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland 4,— DM bzw. für das Ausland 6,— DM Porto und Verpackung.

Bitte benutzen Sie die Bestellkarte.

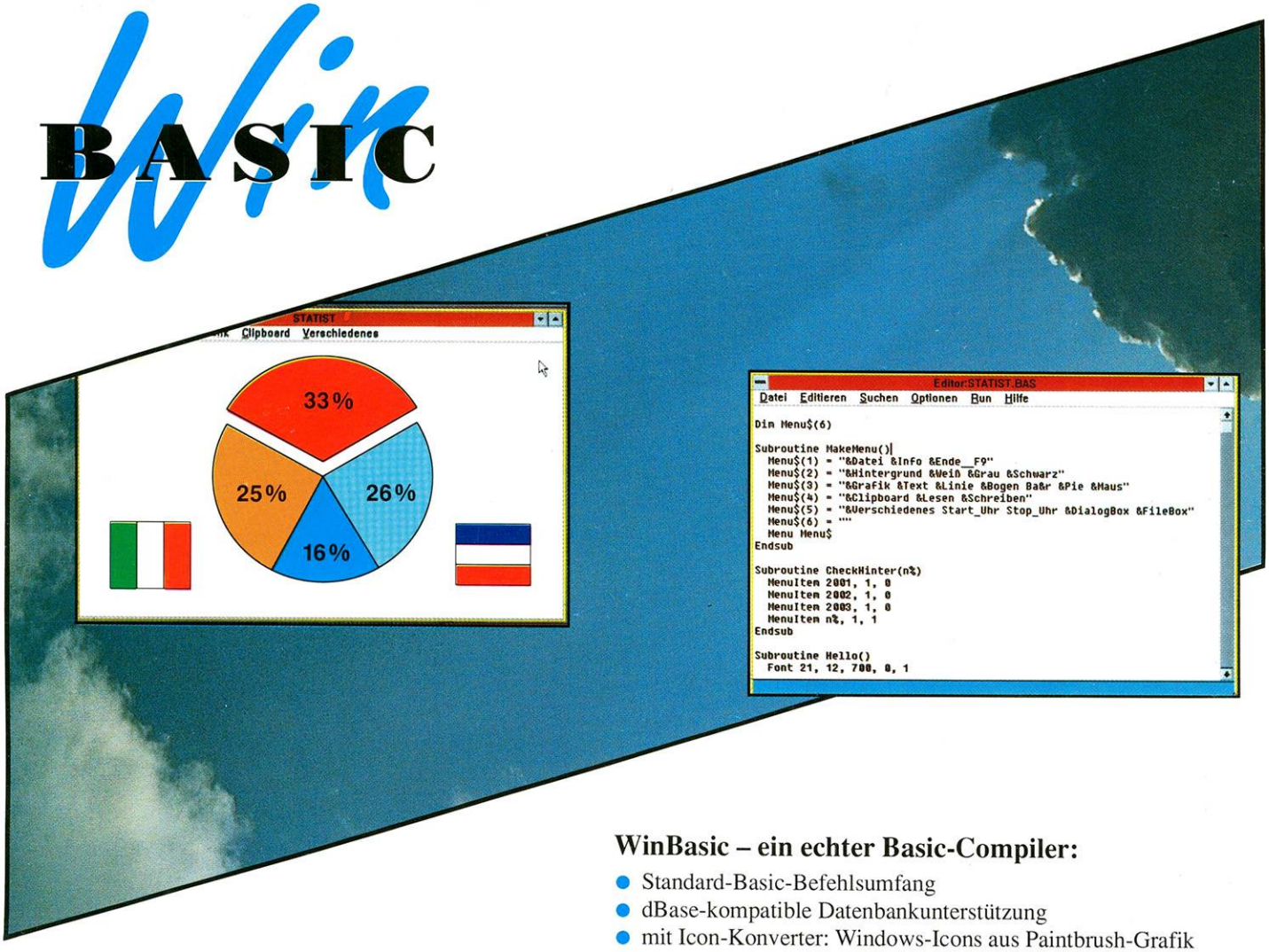
DMV-Verlag · Postfach 250 · 3440 Eschwege

DMV Daten- und Medienverlag

Für Microsoft Windows 2.x und 3.

Der einfache Weg zur Windows-Programmierung

WinBASIC



Kein Windows-Software-Development-Kit nötig! Einfach Windows und WinBasic laden, und schon kann's losgehen!

WinBasic – ein echter Basic-Compiler:

- Standard-Basic-Befehlsumfang
- dBase-kompatible Datenbankunterstützung
- mit Icon-Konverter: Windows-Icons aus Paintbrush-Grafik

WinBasic unterstützt alle wichtigen Windows-Funktionen:

Maus • Fenster • Dialogboxen • Dateiauswahlboxen • EMS und virtueller Speicher • Fonts • Multitasking • Clipboard-Datenaustausch

WinBasic, Basic unter Microsoft Windows **DM 299,—***

Zur Entspannung und als Konzentrationstraining

MEMO, Das Gesellschaftsspiel für helle Köpfe **DM 69,—***

* Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland DM 4,— bzw. für das Ausland DM 6,— Porto und Verpackung. Bitte benutzen Sie die Bestellkarte.

DMV-Verlag • Postfach 250 • 3440 Eschwege

DMV Daten- und Medienverlag