

Frank Kremser
Jörg Koch

AMIGA PROGRAMMIER- HANDBUCH

Für Amiga 500, 1000 und 2000

- ★ *Die wichtigsten Systembibliotheken* ★ *Beispiele für den Aufruf der Betriebssystem-Routinen unter C*
- ★ *Aufruf der DOS-Funktionen* ★ *Programmieren von Windows, Screens und Gadgets* ★ *Grafik und Animation*
- ★ *Tips und Tools in C*

Auf 3 1/2"-Diskette enthalten:
Alle C-Beispiele und Bibliotheken.



Amiga-Programmier-Handbuch

Frank Kremser · Jörg Koch

AMIGA

PROGRAMMIER- HANDBUCH

Für Amiga 500, 1000 und 2000

Die wichtigsten Systembibliotheken
Beispiele für den Aufruf der Betriebssystem-Routinen unter C
Aufruf der DOS-Funktionen
Programmieren von Windows, Screens und Gadgets
Grafik und Animation
Tips und Tools in C

Markt&Technik Verlag AG

Kremser, Frank:

AMIGA-Programmier-Handbuch : Für Amiga 500, 1000 und 2000 ; d. wichtigsten Systembibliotheken ;
Beispiele für d. Aufruf d. Betriebssystem-Routinen unter C ; Aufruf d. DOS-Funktionen ;
Programmieren von windows, screens u. gadjets ;
Grafik u. Animation ; Tips u. tools in C / Frank Kremser ; Jörg Koch. –
Haar bei München : Markt-und-Technik-Verlag, 1987. – & 1 Diskette
ISBN 3-89090-491-2
NE: Koch, Jörg:

Die Informationen im vorliegenden Buch werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.

Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.

Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen.

Trotzdem können Fehler nicht vollständig ausgeschlossen werden.

Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische
Verantwortung noch irgendeine Haftung übernehmen.

Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien.

Die gewerbliche Nutzung der in diesem Buch gezeigten Modelle und Arbeiten ist nicht zulässig.

»Commodore-Amiga« ist eine Produktbezeichnung der Commodore Büromaschinen GmbH, Frankfurt,
die ebenso wie der Name »Commodore« Schutzrechte genießt. Der Gebrauch bzw. die Verwendung bedarf der Erlaubnis
der Schutzrechtsinhaberin.

Amiga ist eine Produktbezeichnung der Commodore-Amiga Inc., USA.

Amiga-BASIC ist eine Produktbezeichnung der Microsoft Inc., USA.

15 14 13 12 11 10 9 8 7 6 5

90 89

ISBN 3-89090-491-2

© 1987 bei Markt&Technik Verlag Aktiengesellschaft,
Hans-Pinsel-Straße 2, D-8013 Haar bei München/West-Germany

Alle Rechte vorbehalten

Einbandgestaltung: Grafikdesign Heinz Rauner

Druck: Schoder, Gersthofen

Printed in Germany

Inhaltsverzeichnis

| | | |
|-------------------------|-------------------------|-----------|
| Vorwort | | 13 |
| Einführung | | |
| 1 | Die Sprache »C« | 15 |
| 1.1 | Datentyp-Umwandlungen | 22 |
| 1.2 | Zeiger | 23 |
| 1.3 | Bedingungen | 24 |
| 1.4 | Schleifen | 26 |
| 1.5 | Strukturen | 27 |
| 1.6 | Die Bibliotheken | 29 |
| 1.7 | Die Devices | 34 |
| Grafikgrundlagen | | |
| 2 | Der Screen | 39 |
| 2.1 | Die View-Modi | 43 |
| 2.2 | Die NewScreen-Structure | 45 |
| 2.3 | Die Screen-Structure | 49 |
| 2.4 | Die Screen-Befehle | 52 |
| 2.4.1 | CloseScreen | 52 |
| 2.4.2 | CloseWorkbench | 52 |
| 2.4.3 | DisplayBeep | 53 |
| 2.4.4 | MakeScreen | 53 |
| 2.4.5 | MoveScreen | 54 |
| 2.4.6 | OpenScreen | 54 |
| 2.4.7 | OpenWorkBench | 55 |
| 2.4.8 | RemakeDisplay | 55 |
| 2.4.9 | RethinkDisplay | 56 |
| 2.4.10 | ScreenToBack | 56 |
| 2.4.11 | ScreenToFront | 56 |
| 2.4.12 | SetRGB4 | 57 |
| 2.4.13 | ShowTitle | 58 |

| | | |
|----------|--------------------|-----------|
| 2.4.14 | WBenchToBack | 58 |
| 2.4.15 | WBenchToFront | 59 |
| 3 | Das Window | 65 |
| 3.1 | Window-Typ | 66 |
| 3.2 | Window-Refreshing | 69 |
| 3.3 | Die Window-Gadgets | 70 |
| 3.4 | IDCMP | 71 |
| 3.5 | Die Window-Befehle | 72 |
| 3.5.1 | ActivateWindow | 75 |
| 3.5.2 | BeginRefresh | 76 |
| 3.5.3 | ClearPointer | 76 |
| 3.5.4 | CloseWindow | 77 |
| 3.5.5 | EndRefresh | 78 |
| 3.5.6 | ModifyIDCMP | 79 |
| 3.5.7 | MoveWindow | 80 |
| 3.5.8 | OpenWindow | 80 |
| 3.5.9 | RefreshWindowFrame | 81 |
| 3.5.10 | ReportMouse | 82 |
| 3.5.11 | SetPointer | 82 |
| 3.5.12 | SetWindowTitles | 84 |
| 3.5.13 | SizeWindow | 85 |
| 3.5.14 | ViewPortAddress | 85 |
| 3.5.15 | WindowLimits | 86 |
| 3.5.16 | WindowToBack | 87 |
| 3.5.17 | WindowToFront | 87 |

Grafikteil

| | | |
|----------|--|-----------|
| 4 | Zeichnen in Screens und Windows | 93 |
| 4.1 | Einfache Zeichenbefehle | 94 |
| 4.1.1 | AreaCircle | 94 |
| 4.1.2 | AreaDraw | 95 |
| 4.1.3 | AreaEllipse | 95 |
| 4.1.4 | AreaEnd | 96 |
| 4.1.5 | AreaMove | 97 |
| 4.1.6 | BNDRYOFF | 97 |
| 4.1.7 | Draw | 97 |
| 4.1.8 | DrawCircle | 98 |
| 4.1.9 | DrawEllipse | 99 |
| 4.1.10 | Flood | 99 |
| 4.1.11 | GetRGB4 | 100 |
| 4.1.12 | InitArea | 100 |

| | | |
|--------|----------------------|-----|
| 4.1.13 | InitTmpRas | 101 |
| 4.1.14 | Move | 102 |
| 4.1.15 | OFF_DISPLAY | 102 |
| 4.1.16 | ON_DISPLAY | 103 |
| 4.1.17 | PolyDraw | 103 |
| 4.1.18 | ReadPixel | 104 |
| 4.1.19 | RectFill | 104 |
| 4.1.20 | ScrollRaster | 105 |
| 4.1.21 | ScrollVPort | 105 |
| 4.1.22 | SetAfPt | 106 |
| 4.1.23 | SetAPen | 107 |
| 4.1.24 | SetBPen | 107 |
| 4.1.25 | SetDrMd | 108 |
| 4.1.26 | SetDrPt | 109 |
| 4.1.27 | SetOPen | 109 |
| 4.1.28 | SetRast | 109 |
| 4.1.29 | SetRGB4 | 110 |
| 4.1.30 | SetWrMsk | 110 |
| 4.1.31 | VBeamPos | 111 |
| 4.1.32 | WaitBOVP | 111 |
| 4.1.33 | WaitTOF | 112 |
| 4.1.34 | WritePixel | 112 |
| 4.2 | Die Textfunktionen | 117 |
| 4.2.1 | AddFont | 117 |
| 4.2.2 | AskFont | 118 |
| 4.2.3 | AskSoftStyle | 119 |
| 4.2.4 | AvailFonts | 119 |
| 4.2.5 | ClearEOL | 121 |
| 4.2.6 | ClearScreen | 121 |
| 4.2.7 | CloseFont | 121 |
| 4.2.8 | OpenDiskFont | 122 |
| 4.2.9 | OpenFont | 123 |
| 4.2.10 | RemFont | 123 |
| 4.2.11 | SetFont | 123 |
| 4.2.12 | SetSoftStyle | 124 |
| 4.2.13 | Text | 125 |
| 4.2.14 | TextLength | 125 |
| 4.3 | Die Images | 129 |
| 4.4 | Umrahmungen: Borders | 132 |
| 4.5 | Intuition-Text | 137 |

| | | |
|----------|--|------------|
| 5 | Einfache Animationen in Screens und Windows | 139 |
| 5.1 | Einfache Hardware-Sprites | 141 |
| 5.1.1 | ChangeSprite | 143 |
| 5.1.2 | FreeSprite | 144 |
| 5.1.3 | GetSprite | 145 |
| 5.1.4 | MoveSprite | 145 |
| 5.1.5 | OFF_SPRITE | 146 |
| 5.1.6 | ON_SPRITE | 146 |
| 5.2 | VSprites | 152 |
| 5.2.1 | AddVSprite | 154 |
| 5.2.2 | DrawGList | 155 |
| 5.2.3 | InitGels | 156 |
| 5.2.4 | LoadView | 157 |
| 5.2.5 | MrgCop | 157 |
| 5.2.6 | InitMasks | 158 |
| 5.2.7 | RemVSprite | 158 |
| 5.2.8 | SortGList | 158 |
| 5.2.9 | WaitTOF | 159 |
| 5.3 | Animation durch SetPointer | 168 |
| 5.4 | Animation durch Preferences | 174 |
| 5.4.1 | GetDefPrefs | 175 |
| 5.4.2 | GetPrefs | 176 |
| 5.4.3 | SetPrefs | 176 |

Programmbedienung

| | | |
|----------|--|------------|
| 6 | Die Programmbedienung | 181 |
| 6.1 | Programmbedienung mit dem Amiga | 182 |
| 7 | Die Menüs | 185 |
| 7.1 | Der Aufbau von Menüs | 186 |
| 7.1.1 | ClearMenuStrip | 188 |
| 7.1.2 | ItemAddress | 189 |
| 7.1.3 | ITEMNUM | 189 |
| 7.1.4 | MENUNUM | 190 |
| 7.1.5 | OffMenu | 190 |
| 7.1.6 | OnMenu | 191 |
| 7.1.7 | SetMenuStrip | 191 |
| 7.1.8 | SUBNUM | 192 |
| 7.2 | Subitems, Command-Tasten, Grafiken und MutualExclude | 193 |
| 7.3 | Die Abfrage von Menüs | 194 |

| | | |
|-------------------------|---|------------|
| 8 | Die Gadgets | 201 |
| 8.1 | Die Gadget-Structure | 203 |
| 8.2 | Das Boolean-Gadget | 210 |
| 8.3 | Das Text/Integer-Gadget | 211 |
| 8.4 | Das Proportional-Gadget | 214 |
| 8.5 | Die Abfrage von Gadgets | 216 |
| 8.6 | Die Gadget-Befehle | 218 |
| 8.6.1 | AddGadget | 218 |
| 8.6.2 | ModifyProp | 218 |
| 8.6.3 | OffGadget | 220 |
| 8.6.4 | OnGadget | 221 |
| 8.6.5 | RefreshGadgets | 221 |
| 8.6.6 | RemoveGadget | 222 |
| | | |
| 9 | System-Meldungen | 231 |
| 9.1 | Die Alerts | 232 |
| 9.1.1 | Der Aufruf von System-Alerts | 233 |
| 9.1.2 | Der Aufruf von Intuition-Alerts | 235 |
| 9.2 | Einfache System-Meldungen durch Requester | 237 |
| 9.3 | Die Requester-Structure | 238 |
| 9.4 | Selbstdefinierte Requester | 241 |
| 9.5 | Das Auto/System-Request | 242 |
| 9.6 | Die Requester-Befehle | 244 |
| 9.6.1 | AutoRequest | 244 |
| 9.6.2 | BuildSysRequest | 245 |
| 9.6.3 | ClearDMRequest | 246 |
| 9.6.4 | EndRequest | 246 |
| 9.6.5 | FreeSysRequest | 246 |
| 9.6.6 | InitRequest | 247 |
| 9.6.7 | Request | 247 |
| 9.6.8 | SetDMRequest | 248 |
| | | |
| Ein- und Ausgabe | | |
| | | |
| 10 | Die Ein- und Ausgabe | 255 |
| 10.1 | DOS-Funktionen in Programmen | 256 |
| 10.1.1 | Close | 257 |
| 10.1.2 | CreateDir | 257 |
| 10.1.3 | CurrentDir | 258 |
| 10.1.4 | DeleteFile | 258 |
| 10.1.5 | DupLock | 259 |
| 10.1.6 | Examine | 259 |
| 10.1.7 | Execute | 260 |

| | | |
|-----------|-------------------------------|------------|
| 10.1.8 | ExNext | 261 |
| 10.1.9 | Info | 261 |
| 10.1.10 | Input | 262 |
| 10.1.11 | IOErr | 262 |
| 10.1.12 | IsInteractive | 262 |
| 10.1.13 | Lock | 263 |
| 10.1.14 | Open | 264 |
| 10.1.15 | Output | 264 |
| 10.1.16 | ParentDir | 265 |
| 10.1.17 | Read | 265 |
| 10.1.18 | Rename | 266 |
| 10.1.19 | Seek | 267 |
| 10.1.20 | SetComment | 268 |
| 10.1.21 | SetProtection | 268 |
| 10.1.22 | Unlock | 269 |
| 10.1.23 | WaitForChar | 269 |
| 10.1.24 | Write | 270 |
| 10.2 | DOS-Demonstration | 271 |
| 11 | Der Drucker | 277 |
| 11.1 | Druckerausgabe über Amiga-DOS | 278 |
| 11.2 | Die printer.device | 279 |

Sonderteil

| | | |
|-----------|--------------------------|------------|
| 12 | Die Workbench | 287 |
| 12.1 | AddFreeList | 288 |
| 12.2 | AllocWLObject | 289 |
| 12.3 | BumpRevision | 290 |
| 12.4 | FreeDiskObject | 291 |
| 12.5 | FreeFreeList | 292 |
| 12.6 | FreeWLObject | 293 |
| 12.7 | GetDiskObject | 294 |
| 12.8 | GetIcon | 295 |
| 12.9 | GetWLObject | 296 |
| 12.10 | PutDiskObject | 297 |
| 12.11 | PutIcon | 298 |
| 12.12 | PutWLObject | 299 |
| 13 | Die Sprachausgabe | 301 |

| | | |
|-----------|-------------------------------|------------|
| 14 | Multitasking | 305 |
| 14.1 | ChangePri | 307 |
| 14.2 | CreateTask | 308 |
| 14.3 | DeleteTask | 309 |
| 14.4 | RemTask | 310 |
| | | |
| 15 | Mathematik-Libraries | 313 |
| 15.1 | Mathematische Grundfunktionen | 314 |
| 15.1.1 | SPFix | 315 |
| 15.1.2 | SPFlt | 315 |
| 15.1.3 | SPCmp | 316 |
| 15.1.4 | SPTst | 316 |
| 15.1.5 | SPAbs | 317 |
| 15.1.6 | SPNeg | 317 |
| 15.1.7 | SPAdd | 317 |
| 15.1.8 | SPSub | 318 |
| 15.1.9 | SPMul | 318 |
| 15.1.10 | SPDiv | 319 |
| 15.2 | Transzendente Funktionen | 320 |
| 15.2.1 | SPAsin | 321 |
| 15.2.2 | SPAcos | 321 |
| 15.2.3 | SPAtan | 321 |
| 15.2.4 | SPSin | 322 |
| 15.2.5 | SPCos | 322 |
| 15.2.6 | SPTan | 323 |
| 15.2.7 | SPSincos | 323 |
| 15.2.8 | SPSinh | 324 |
| 15.2.9 | SPCosh | 324 |
| 15.2.10 | SPTanh | 324 |
| 15.2.11 | SPExp | 325 |
| 15.2.12 | SPLog | 325 |
| 15.2.13 | SPLog10 | 326 |
| 15.2.14 | SPPow | 326 |
| 15.2.15 | SPSqrt | 327 |
| 15.2.16 | SPTieee | 327 |
| 15.2.17 | SPFieee | 327 |
| | | |
| 16 | Das IFF-Bild-Format | 331 |
| | | |
| 17 | Sonstige Befehle | 341 |
| 17.1 | CurrentTime | 342 |
| 17.2 | DoubleClick | 343 |

Anhang

| | | |
|----------|--|------------|
| A | Zuweisungen Befehle <-> Libraries | 345 |
| B | Die Structures | 355 |
| C | System Alerts | 367 |
| D | Die DOS-Fehlermeldungen | 371 |
| E | Anmerkungen zur Programmgestaltung | 375 |
| F | Drucker-Codes | 377 |
| G | Die Demo-Diskette | 379 |
| | Stichwortverzeichnis | 383 |
| | Hinweise auf weitere Markt&Technik-Produkte | |

Vorwort

Der Commodore Amiga ist ein vielseitiger Computer. Somit fällt es vielen sehr schwer, diesen Rechner in die richtige Kategorie einzuordnen. Seine Anwendungsgebiete reichen vom professionellen Einsatz als Personalcomputer im Business-Bereich bis zum idealen Rechner für Programmierer.

Unter anderem setzt sich, nicht zuletzt des Amiga wegen, immer mehr eine neue Art der Programmgestaltung und der Benutzeroberfläche durch. Die alten, nicht bedienungsfreundlichen Programme werden nun bald der Geschichte angehören. Umständliche Tastaturbedienung von Programmen, Schwarzweiß-Grafiken oder solche mit vier Farben, Grafiken mit niedriger Auflösung, langsame Geschwindigkeiten und ein öder Ablauf von Programmen zählen nun, dank Amiga, zur Vergangenheit. Schnelle Grafiken und hohe Rechengeschwindigkeiten, Mausbedienung sowie Multitasking beherrschen nun das Geschehen auf dem Software-Markt, nach dem sich jeder Hobby- oder Profiprogrammierer richten muß, wenn seine Software großen Zuspruch bei den Software-Anwendern finden soll.

Wir persönlich sehen im Amiga eine faszinierende Maschine für Programmierer, die sehr viele Reize und Geheimnisse enthält. Diese Reize und Geheimnisse wollen wir in diesem Buch dem Leser anhand von vielen Beispielen verdeutlichen. Gerade diese Programmbeispiele sehen wir in diesem Buch als etwas ganz Besonderes an. Es ist doch oft so, daß viel erklärt wird, aber ein Beispiel, das noch offene Fragen beantwortet, fehlt. Genau dieses Übel wollten wir mit den vielen Demonstrationsprogrammen beseitigen. Wir raten also jedem Leser, wenn er etwas nicht versteht, in dem zugehörigen Programm nachzusehen und es genau durchzuarbeiten. Da alle Programme auf Diskette mitgeliefert werden, als Source-Code und kompiliert, kann auch gleichzeitig überprüft werden, was das Programm überhaupt macht.

Schon oft haben wir uns ein kompaktes Buch wie dieses gewünscht, das dem Programmierer nicht nur zeigt, was der Rechner kann, sondern ihm auch an Beispielen die Anwendung der Features verdeutlicht. Die Entwicklung von Software der neuesten Generation dürfte somit kein Problem mehr sein. Leider können wir aber nicht alle Möglichkeiten des Amiga auführen und erklären, da dazu zwei weitere derartige Bücher erforderlich wären. Da allerdings sehr viele Funktionen durch andere, kompaktere Befehle ersetzt werden können, beschränken wir uns auf diese, was aber keineswegs große Einschränkungen in der Programmierung mit sich bringt. Im Gegenteil, die Programmierung wird dadurch sogar erheblich vereinfacht. In diesem Buch sind alle Funktionen beschrieben, die der Programmierer für ein sehr gutes Programm benötigt.

Da heutzutage Leistungsfähigkeit und hohe Programmgeschwindigkeiten von großer Bedeutung sind, hielten wir es nicht für angebracht, die Demo-Programme in BASIC zu schreiben, obwohl das Amiga-BASIC zu den Leistungsfähigsten seiner Klasse zählt. Da sich bei der Entwicklung von Programmen und bei den Programmierern zunehmend die Programmiersprache C durchsetzt, wollten auch wir natürlich nicht darauf verzichten. Programmierer, die kaum Erfahrungen mit C besitzen, brauchen aber nun nicht zurückzuschrecken, da die Listings dokumentiert und mehrfach von uns bearbeitet worden sind, so daß wir garantieren können, daß alle Programme auf einem Amiga-System mit mindestens 512 Kbyte laufen.

Danken möchten wir nun noch Frau Christine Baumann für die tatkräftige Unterstützung und unseren Eltern für das Verständnis, das sie aufbrachten, wenn wir nächtelang am Computer saßen, um doch noch einige Verbesserungen in das Buch einzubauen. Auch Herrn Peter Wollschlaeger, der sich unseres Buches als Lektor annahm, gehört unser Dank.

Die Sprache »C«

Auf dem Amiga hat sich die Sprache »C« als Programmiersprache für anspruchsvolle Programme durchgesetzt. Dies liegt vornehmlich daran, daß C-Programme enorm schnell, dabei aber dennoch recht einfach zu erstellen sind.

Für die C-Anfänger gehen wir in diesem Kapitel auf die Programmierung in C ein. Natürlich können nicht alle Einzelheiten dargestellt werden, da dies den Rahmen des Buches sprengen würde. Als Einstieg dürften die Informationen allerdings ausreichend sein.

Für den Amiga stehen derzeit zwei verschiedene C-Compiler zur Verfügung. Dies sind der Manx-Aztec- und der Lattice-C-Compiler. Der Unterschied liegt im Preis, bei der Bedienungsfreundlichkeit und in der Schnelligkeit und Kompaktheit des erzeugten Codes. Beide lassen sich leider nur vom CLI, dem »Command Line Interface« des Amiga, bedienen, was anfangs sehr mühsam ist. Wir möchten Ihnen an dieser Stelle keinen Compiler anraten, da dies schon in großer Zahl in den Fachzeitschriften geschehen ist.

Die Programme haben wir auf dem Lattice-Compiler V3.10 geschrieben. Den Besitzern von älteren Versionen des Lattice-C-Compilers stehen leider noch nicht alle Befehle zur Verfügung, beispielsweise sind die Befehle AreaCircle und AreaEllipse noch nicht implementiert. Trotz der Verwendung des Lattice-Compilers sollten die Programme alle auch mit dem Aztec-Compiler lauffähig sein. Inkompatibilitäten wurden uns bisher jedenfalls nicht bekannt. Wenn Sie Besitzer der Lattice-Version 4.00 sein sollten, so sind die Programme auch mit dieser Version lauffähig, wenn Sie zum Kompilieren das später angeführte Batch-File verwenden.

Zudem möchten wir jedem Programmierer dazu raten, eine Festplatte zu verwenden, sei es eine Amiga- oder eine PC-Harddisk im SideCar bzw. Amiga 2000, da das Kompilieren mit Disketten äußerst zeitraubend ist, zumal diese für C eine nicht gerade üppige Speicherkapazität besitzen, so daß es sehr schnell zu Speichermangel auf den Disketten kommen kann.

Zur Vereinfachung des Kompilervorganges haben wir ein Batch-File geschrieben, das alle Kompilier- und Linkphasen selbständig durchführt:

Für die Lattice-Version 3.02 oder 3.03:

```
stack 20000
if not exists <prg>.c
    echo "File ist nicht vorhanden"
    skip end
endif
echo "-- kompilieren --"
lc1 -i:include/ -i:include/lattice/ <prg>.c
if not exists <prg>.q
    echo "Compiler-Fehler"
    quit 20
endif
lc2 -cdb <prg>
alink : lib/lstartup.obj+<prg>.o library : lib/lc.lib+
:lib/amiga.lib to <prg> map nil:
delete <prg>.o
echo "-- Kompilier- und Linkvorgang ist zu Ende --"
lab end
```

Für die Lattice-Version 3.10:

```
stack 20000
if not exists <prg>.c
    echo "File ist nicht vorhanden"
    skip end
endif
echo "-- kompilieren --"
LC1 -f -i:include/ -i:include/lattice/ <prg>.c
if not exists <prg>.q
    echo "Compiler-Fehler"
    quit 20
endif
LC2 -cdb <prg>
BLINK FROM LIB:c.o+<prg>.o TO <prg> LIB LIB:lc.lib+lcmffp.lib+
LIB:amiga.lib+LIB:lcm.lib
delete <prg>.o
echo "-- Kompilier- und Linkvorgang ist zu Ende --"
lab end
```

Für die Lattice-Version 4.00:

```
stack 20000
if not exists <prg>.c
    echo "File ist nicht vorhanden"
    skip end
endif
echo "-- kompilieren --"
LC1 -f -i:include/ -i:include/lattice/ <prg>.c
if not exists <prg>.q
    echo "Compiler-Fehler"
    quit 20
endif
LC2 <prg>
BLINK FROM LIB:c.o+<prg>.o TO <prg> LIB LIB:lc.lib+lcmffp.lib+
LIB:amiga.lib+LIB:lcm.lib
delete <prg>.o
echo "-- Kompilier- und Linkvorgang ist zu Ende --"
lab end
```

Dieses File muß mit dem Editor »ed« eingegeben und gespeichert werden. Dazu gehen Sie von der Workbench aus in das CLI. Dort erscheint »1>«. Nun kommen Sie mit »ed comp« in den Editor und können das Batch-File eingeben. Wenn Sie fertig sind, können Sie es abspeichern, indem Sie »ESC« und anschließend »x« drücken. Das Batch-File steht nun unter dem Namen »comp« auf Ihrer Diskette bzw. Harddisk. Wenn Sie später ein selbstgeschriebenes C-Programm kompilieren wollen, starten Sie es mit dem CLI-Befehl »EXECUTE comp« und dem Programmnamen des C-Programms ohne das »c«-Kürzel.

Nun, was bewirkt dieses Batch-File? Zu Beginn wird der Programmname der Variablen »prg« übergeben und anschließend der Stack auf eine Größe von 20000 Byte gesetzt, was nötig ist, da der Compiler eine Vielzahl von Daten zwischenspeichern muß. Anschließend überprüft es, ob das gewünschte Programm zum Kompilieren überhaupt existiert. Ist das Programm nicht vorhanden, steigt das Batch-File aus und druckt die Fehlermeldung »File ist nicht vorhanden«. Ist kein Fehler aufgetreten, so beginnen nun die Kompilierungsvorgänge »lc1« und »lc2«. »lc1« überprüft hauptsächlich die Syntax des Hauptprogramms und der eingeladenen Include-Files. »lc2« generiert anschließend den Programmcode. Tritt beim Kompilierungsvorgang »lc1« ein Fehler auf, so wird auch hier der Ablauf des Batch-Files gestoppt und eine Fehlermeldung »Compiler-Fehler« ausgegeben.

Liefen jedoch die Kompilierungsvorgänge ohne Fehler ab, beginnt das Programm mit dem Zusammenfügen, sprich »Linken«, der Bibliotheksmodule mit dem Programmcode. Eine Meldung teilt dem Benutzer anschließend mit, daß dieser Prozeß beendet ist. Danach kann das »kompilierte« und »gelinkte« Programm gestartet werden. Das Programm muß unter dem gewünschten Namen mit angehängtem »c« erstellt und abgespeichert worden sein, also beispielsweise »test.c«. Nach dem Kompilier- und Linkvorgang steht der startbare Programmcode in der Datei »test«. Dieser Programmcode läßt sich nun einfach durch Eintippen des Dateinamens starten.

Bei den meisten Programmen bietet sich auch noch die Möglichkeit an, eine »info«-Datei zu kopieren, beispielsweise »copy cli.info to test.info«. Dann können die Programme auch von der Workbench aus gestartet werden. Eine Ausnahme bilden hier die Programme, die »printf«, »scanf« oder DOS-Befehle benutzen, da diese Befehle ihre Ein- und Ausgabe über CLI abwickeln. Aus diesem Grund besitzen auch nicht alle Demonstrationsprogramme auf der mitgelieferten Diskette sog. Icons, sind also von der Workbench aus nicht sichtbar oder startbar. Sie können nur von CLI aus gestartet werden. Verwenden Sie allerdings das zuvor beschriebene »c.o.«, so können Sie auch diese von der Workbench starten! Hier nun auch noch ein Batch-File für die Besitzer eines Aztec-Compilers:

```
.key prg
cc -t<prg>.c+1
as <prg>.asm
ln <prg>.o -lm -lc
echo "Kompilier-, Assembler- und Linkvorgang ist zu Ende"
```

Für alle nachfolgenden Erklärungen möchten wir Sie bitten, alle Schritte direkt am Computer nachzuvollziehen, da es dann leichter für Sie wird. Zu Beginn müssen Sie natürlich den Computer starten. Falls Sie es nicht schon zuvor getan haben, müssen Sie das Preference-Programm starten und den CLI-Schalter auf »ON« setzen, da nur in diesem Fall CLI zu verwenden ist. Anschließend können Sie wieder Preference verlassen, am besten mit »Save«, da dann das CLI-Icon auch nach dem Einschalten des Computers erscheint. Nun starten Sie bitte CLI durch einen Doppelklick auf das CLI-Icon. Kurz darauf erscheint das CLI-Window. Ist es das einzige CLI-Window auf dem Bildschirm, so müsste das »1>«-Prompt darin erscheinen. Dahinter ist der Cursor zu erkennen. Nun können Sie sämtliche CLI-Befehle verwenden. Als Beispiel dafür tippen Sie bitte »dir« ein - mit anschließendem »RETURN«. Sie sehen nun das Inhaltsverzeichnis der Hauptdiskette.

Nun wollen wir mit der Einführung in »C« beginnen. C ähnelt in vielen Punkten den Programmiersprachen Pascal und Modula, weshalb Pascal- und/oder Modula-Programmierer keine Schwierigkeit haben dürften, auf C umzusteigen.

»C« wurde 1972 in den USA entwickelt, 1973/74 verbessert und anfangs vornehmlich unter dem Betriebssystem UNIX verwendet. Da diese Sprache möglichst flexibel sein sollte, wurden ihr nur sehr wenige Befehle fest implementiert. Darunter sind:

| | |
|--------------|---------------------------------|
| - if | -> Bedingte Anweisung |
| - switch | -> Bedingte Anweisungen |
| - for, while | -> Zähl- und bedingte Schleifen |

Es sind noch einige Befehle mehr vorhanden, auf die wir allerdings nicht eingehen werden, da sie nicht von größerer Bedeutung sind. Diese Befehle genügen jedoch, um alle programmkontrollierenden Funktionen durchführen zu können, zumal sich die meisten Befehle in Include-Dateien oder in Bibliotheken befinden, die der Sprache C zu ihrer Leistungsfähigkeit und Flexibilität verhelfen.

Ein C-Programm setzt sich normalerweise aus drei Grundteilen zusammen. Im ersten Teil gibt der Programmierer an, welche Include-Dateien oder Bibliotheken er verwenden will, die somit beim Kompilieren eingelesen werden müssen. Im zweiten Teil werden die globalen Variablen deklariert. Auf diese Variablen kann von jeder Routine des Programms aus zugegriffen werden. Der dritte Teil besteht aus dem eigentlichen Programm. Dieser Teil gliedert sich allerdings wieder in zwei Teile auf. Da ist zum einen der Teil mit den

Unterroutinen - in Pascal auch Procedures oder Functions genannt - und zum anderen der sogenannte »main«-Teil. Dieser Teil stellt die Hauptroutine des Programms dar, die beim Start des Programms aktiviert wird. Der Unterschied zu Pascal besteht darin, daß Unterroutinen und Hauptprogramm keine festgelegte Reihenfolge haben müssen. Es können in C also auch Unterroutinen aufgerufen werden, die erst später im Programm folgen.

C-Programme wirken leider teilweise sehr undurchsichtig, was aber durch das Einfügen von Kommentaren und Unterroutinen wettgemacht wird. Solche Kommentare werden mit »/*« und »*/« geklammert. Aufpassen sollte man auf solche »Kleinigkeiten« wie Semikolons oder Kommata, da der Compiler oftmals solche Fehler nicht erkennt, sondern weiterkompiliert, was zum Absturz beim späteren Starten des Programms führen kann. Mit »C« zu arbeiten, heißt also korrekt und sauber arbeiten, sonst kann für nichts garantiert werden.

Zusätzlich zu den oben genannten Befehlen besitzt »C« noch eine Reihe von Funktionen, von denen »printf« und »scanf« die wichtigsten sind. »printf« dient zur Ausgabe von Texten, Zahlen u.a. Mittels »scanf« können Texte und Zahlen eingelesen werden.

Zu der Funktion »printf« wollen wir an dieser Stelle ein erstes Programm erstellen. Es soll nur den Text »Mein erstes C-Programm ausgeben«. Da Sie sich schon in CLI befinden, müssen Sie nur noch den Editor »ed« aktivieren. Zusätzlich muß noch der Name angegeben werden, den unser Programm haben soll, gefolgt von ».c«:

```
1> ed test.c
```

Nun befinden Sie sich im Editor. Da wir nur den Befehl »printf« verwenden wollen, der von »C« bereitgestellt wird, müssen wir keine Include-Dateien einlesen oder Variablen deklarieren. Also können wir gleich mit dem Programm beginnen. Das Programm besteht in unserem Fall nur aus der Hauptroutine, die mit »main()« eingeleitet wird. Anschließend folgen die Anweisungen, die zur Hauptroutine gehören. Sie müssen mit den zwei geschweiften Klammern »{« und »}« geklammert werden. Zwischen diesen Klammern steht also das Hauptprogramm. In unserem Fall besteht es nur aus »printf("Mein erstes C-Programm\n");«. »printf« gibt, wie schon zuvor erwähnt, einen Text aus. Der Text steht anschließend in Klammern und von Hochkommata eingegrenzt. Ein Sonderfall ist noch mit eingebaut: »\n« bewirkt einen Zeilenvorschub, was nötig ist, da »printf« keinen automatischen Zeilenvorschub bewirkt. Neben »\n« gibt es unter anderem noch »\0«, was den ASCII-Code Null darstellt. Nun sieht unser Programm also folgendermaßen aus:

```
main()
{
  printf("Mein erstes C-Programm\n");
}
```

Betätigen Sie nun die »ESC«-Taste und anschließend »X« und »RETURN«, um das Programm unter dem Namen »test.c« zu speichern. Um das Programm nun starten zu können, muß es zuerst mit dem Compiler in Maschinensprache übersetzt werden. Dies kann mit dem oben angegebenen Batch-File geschehen:

```
1> EXECUTE comp test
```

Nach einer Weile ist der Kompilier-Vorgang beendet, und es erscheint wieder das »1>«-Prompt. Nun kann das Programm mittels der Eingabe von »test« mit anschließendem »RETURN« gestartet werden. Das Ergebnis ist zwar nicht aufregend, aber es zeigt doch die Vorgehensweise beim Erstellen eines C-Programms.

Nun wollen wir einen Schritt weitergehen, wir wollen weitere Funktionen verwenden, die sich in einer Include-Datei auf der Diskette befinden. Diese Funktionen sind »getchar« und »putchar«, die sich in der Datei »stdio.h« befinden. Zusätzlich deklarieren wir zwei Variablen. Die Variable »global« kann in allen Routinen verwendet werden. »lokal« kann nur in der Hauptroutine verwendet werden. Wäre eine weitere Routine vorhanden, so könnten in ihr nur die Variable »global« sowie ihre eigenen lokalen Variablen verwendet werden.

»getchar« und »putchar« haben im Prinzip die gleiche Funktion wie »scanf« und »printf«, sind jedoch für einzelne Zeichen ausgelegt.

Um das Programm erstellen zu können, müssen Sie als erstes in den Editor mit »ed test2.c«. Das Programm sieht dann folgendermaßen aus:

```
#include <stdio.h>

char global;

main()
{
  char lokal;
  lokal = getchar();
  global = lokal;
  putchar(global);
}
```

Auch dieses Programm muß nach dem Speichern kompiliert werden. Dies geschieht mit »EXECUTE comp test2«.

An dieser Stelle wollen wir auf die möglichen Datentypen eingehen, mit denen Variablen deklariert werden können:

| Datentyp | Wertebereich | | Speicherlänge |
|--------------|---------------------------|----------------------------|---------------|
| int | - -32768 | bis 32767 | - 2 BYTE |
| long int | - $-2 \cdot 10$ hoch 9 | bis $2 \cdot 10$ hoch 9 | - 4 BYTE |
| unsigned int | - 0 | bis 65535 | - 2 BYTE |
| char | - 0 | bis 255 (ASCII) | - 1 BYTE |
| FLOAT | - ≈ 10 hoch -37 | bis ± 10 hoch 38 | - 4 BYTE |
| DOUBLE | - ≈ 10 hoch -307 | bis ± 10 hoch 308 | - 8 BYTE |
| /UBYTE | - 0 | bis 255 | - 1 BYTE |
| /UWORD | - 0 | bis 65535 | - 2 BYTE |
| /ULONG | - 0 | bis $4,3 \cdot 10$ hoch 9 | - 4 BYTE |
| BYTE | - -128 | bis 127 | - 1 BYTE |
| WORD | - -32768 | bis 32767 | - 2 BYTE |
| LONG | - $-2,15 \cdot 10$ hoch 9 | bis $2,15 \cdot 10$ hoch 9 | - 4 BYTE |

Nun wollen wir noch ein Programm schreiben, das Unterrouninen verwendet. Solchen Routinen können Parameter übergeben werden, sie können aber auch Werte zurückgeben. Hier das Programm:

```
#include <stdio.h>

char eingabe;

routine(wert)                /* Routinenkopf mit Parameter */
char wert;                  /* Datentyp des Parameters */
{
    putchar(wert);
    eingabe = getchar();
    return(eingabe);
}

main()                      /* Hauptroutine */
{
    eingabe = getchar();
    eingabe = routine(eingabe); /* Routine aufrufen */
    putchar(eingabe);
}
```

Nun wissen Sie über die Struktur von C-Programmen Bescheid. In den nachfolgenden Teilen gehen wir näher auf die Programmierung ein.

1.1 Datentyp-Umwandlungen

C bietet die Möglichkeit, Daten innerhalb des Programms auf einfache Weise in andere Datentypen umzuwandeln. Dazu braucht nur vor den umzuwandelnden Wert bzw. vor die Variable in runden Klammern der Datentyp gesetzt zu werden, in den umgewandelt werden soll. Wenn beispielsweise eine Integerzahl in einer Integervariablen gespeichert ist, aber einer FLOAT-Variablen zugewiesen werden soll, so geschieht das folgendermaßen:

```
floatvar = (FLOAT) intvar;
```

Dies gilt auch für Structures, die später erläutert werden. Als Besonderheit gilt an dieser Stelle, daß in solchen Fällen das Wort »struct« noch davor gesetzt werden muß. Beispiel:

```
struct test1    /* erste Structure deklarieren */
{
    FLOAT float;
    int  int;
};

struct test2    /* zweite Structure deklarieren */
{
    FLOAT float;
    int  int;
};

main()
{
    struct test1 var1; /* Variablen vom Typ test1 */
    struct test2 var2; /* und test2 deklarieren */
    .
    .
    var2 = (struct test2) var1; /* Zuweisen und umwandeln, */
    .                               /* da nicht vom gleichen Typ */
    .
}
```

1.2 Zeiger

Ein sehr wichtiges Thema sind die Zeiger. Sie gibt es zwar auch in Pascal und Modula, doch ist ihre Verwendung in C besonders flexibel.

Ein Zeiger, auch Pointer oder Ptr genannt, ist eine Adreßvariable. Die Adresse, die sie enthält, ist das erste Byte einer Variablen. Man sagt auch, der Zeiger zeigt auf die Variable.

Deklarieren kann man einen Zeiger folgendermaßen:

```
FLOAT *flt;
```

Wir haben also einen Zeiger auf eine Float-Variable deklariert. Durch den Stern »*« wird »flt« zum Zeiger. Es muß aber beachtet werden, daß durch diese Deklaration nur Speicherplatz für den Zeiger, nicht aber für die Variable bereitgestellt wird.

Zu den Zeigern gehört auch der Adreßoperator »&«. Er ermittelt die Adresse einer Variablen. Das bedeutet, wenn man einen Zeiger auf eine bestimmte Variable setzen will, so geht man folgendermaßen vor:

```
FLOAT *flt; /* Zeiger deklarieren */
FLOAT var; /* Variable deklarieren */

flt = &var; /* Zeiger auf Variable setzen */
```

Durch den Adreßoperator kann man also die Adresse einer Variablen ermitteln. Umgekehrt kann durch den Stern »*« auf den Speicherbereich zugegriffen werden, auf den *flt zeigt:

```
FLOAT *flt; /* Zeiger deklarieren */
FLOAT var; /* Variable deklarieren */

*flt = var; /* Variablenwert in Bereich kopieren,
             auf flt zeigen */
```

1.3 Bedingungen

In der Sprache C sind verschiedene Möglichkeiten vorhanden, zu testen, ob eine Bedingung wahr oder falsch ist.

Die erste Möglichkeit ist die Verwendung des »if«-Befehls. Er hat folgende Syntax:

```
if(BEDINGUNG)
    DANN;
else
    ANSONSTEN;
```

Alle kleingeschriebenen Wörter sind in dieser Form anzugeben. Auf den »else«-Zweig kann verzichtet werden. »DANN« gibt den Befehl bzw. die Befehle an, die ausgeführt werden sollen, wenn die Bedingung wahr ist. Wenn nur ein Befehl ausgeführt werden soll, so wird dieser normal angegeben:

```
if(x == y)
    printf("x ist gleich y");
```

Sollen mehrere Befehle ausgeführt werden, so müssen diese geklammert werden:

```
if(x == y)
{
    printf("x ist gleich y,\n");
    printf("also ist y auch gleich x");
};
```

Das gleiche gilt auch für den »else«-Zweig.

Die Bedingung besteht immer aus dem Vergleich zweier Werte miteinander. Folgende Vergleichsoperatoren stehen zur Verfügung:

| | | |
|----|---|-------------------------|
| > | - | Größer als |
| >= | - | Größer als oder gleich |
| < | - | Kleiner als |
| <= | - | Kleiner als oder gleich |
| == | - | Gleich |
| != | - | Nicht Gleich |

Wenn x gleich 4 und y gleich 7 ist, dann ist also die Bedingung (x != y) wahr, da x ungleich y ist.

Eine weitere Möglichkeit, einen Programmteil nur unter bestimmten Bedingungen ablaufen zu lassen, stellt der Befehl »switch« dar. Er hat folgende Syntax:

```
switch(AUSDR)
{
  case AUSDR1 : DANN;
  case AUSDR2 : DANN;
  .
  .
  usw.
};
```

AUSDR ist ein Wert, der mit den Ausdrücken nach den »case«-Marken verglichen wird. Sind dann beide Werte gleich, so wird der Befehl bzw. werden die Befehle nach dem Doppelpunkt ausgeführt.

1.4 Schleifen

Wiederholungen innerhalb eines Programms nennt man Schleifen. C kennt verschiedene Arten von Schleifen. Der erste Typ ist die Zählschleife:

```
for(INIT; BED; INC)
    BEFEHLE;
```

Bei INIT muß die Schleifenvariable, die für die Zählschleife benötigt wird, auf den Anfangswert gesetzt werden. Diese Schleifenvariable muß von einem ganzzahligen Typ sein.

BED stellt die Abbruchbedingung der Schleife dar. Wenn diese Bedingung nicht mehr erfüllt ist, wird die Schleife abgebrochen. Welche Bedingungen möglich sind, können Sie aus Kapitel 1.3 ersehen.

Da die Schleifenvariable nicht automatisch erhöht oder erniedrigt wird, müssen Sie selbst diese Aufgabe übernehmen. Dies geschieht bei INC.

Als Beispiel führen wir nun eine Zählschleife an, die von 0 bis 1000 zählt:

```
int zaehler;

for(zaehler = 0; zaehler <= 1000; zaehler++)
    printf("\n%d", zaehler);
```

»zaehler + +« hat die gleiche Bedeutung wie »zaehler = zaehler + 1«.

Eine weitere Möglichkeit, Schleifen zu bilden, ist die While-Schleife. Sie wird solange durchlaufen, bis die Bedingung nicht mehr gilt. Ihre Syntax:

```
while(BEDINGUNG)
    BEFEHL;
```

Die Bedingung entspricht der der If-Anweisung.

Ähnlich wie die While-Schleife funktioniert die »do..while«-Schleife. Die Besonderheit liegt darin, daß das Abbruchkriterium erst nach einmaligem Durchlaufen der Befehle innerhalb der Schleife geprüft wird:

```
do
    BEFEHL;
while(BEDINGUNG);
```

Auch in diesem Fall müssen die Befehle innerhalb der Schleife geklammert werden, wenn die Schleife aus mehr als einem Befehl besteht.

1.5 Strukturen

Von grundlegender Bedeutung sind die Strukturen, auch Structures oder Listen genannt. In ihnen können verschiedene Variablen unter einem Oberbegriff zusammengefaßt werden. Eine Structure, wie wir sie nachfolgend nennen wollen, wird von dem Wort »struct« eingeleitet. Ihm folgt der Name, den die Structure haben soll, gefolgt von den Einträgen, die in der Structure zusammengefaßt werden sollen. Solche Einträge können auch weitere Structures sein. Ein Beispiel:

```
struct Bsp
{
    FLOAT     flt;
    int       i;
    struct Test demostruct;
    struct Bsp *ptr;
};
```

Die Structure Bsp besteht also aus den Einträgen flt, i, demostruct und dem Zeiger ptr, der auf eine weitere Structure vom Typ Bsp zeigt.

Bsp stellt nun einen neuen Datentyp dar. Um ihn verwenden zu können, muß eine Variable von diesem Typ deklariert werden. Dies geschieht folgendermaßen:

```
struct Bsp beispiel;
```

Will man nun auf die einzelnen Einträge zugreifen, so geschieht das folgendermaßen:

Auf flt, i und demostruct kann sehr einfach zugegriffen werden:

```
beispiel.flt = 2.45;
beispiel.i   = 3;
```

also einfach durch einen Punkt zwischen dem Variablennamen und dem Eintrag, auf den zugegriffen werden soll.

Um auf die Float-Zahl der Structure zugreifen zu können, auf die ptr zeigt, muß ein »Pfeil« zwischengesetzt werden. Dieser »Pfeil« ist eigentlich nur eine Abkürzung für die ausführliche Schreibweise »(*beispiel.ptr).flt«:

```
beispiel.ptr->flt = ....;
```

Man muß nicht unbedingt eine Variable vom diesem Structuretyp deklarieren. Es kann auch ein Zeiger darauf verwendet werden, für den die gleichen Bedingungen gelten wie für die Zeiger in Kapitel 1.2. Ein solcher Zeiger wird folgendermaßen deklariert:

```
struct Bsp *beispiel;
```

Nun noch einige allgemeine Anmerkungen zu diesem Buch:

Beim Erstellen der Programme sind wir von der Lattice C Version ausgegangen. Zur korrekten Einstellung des Screens haben wir ein kleines C – Programm geschrieben, das ein Test-Bild in einer Auflösung von 640 x 512 Pixels erzeugt. Dieses Bild muß den gesamten Bildschirm ausfüllen. Die Regler für die Vertikal bzw. Horizontalablenkung finden Sie am hinteren Teil des Monitors.

Das »Testbild«-Programm öffnet einen Screen mit einer Auflösung von 640 x 512 Pixels. Anschließend wird je eine rote, grüne und blaue Box in die obere Hälfte, sowie eine schwarze Box mit einem weißem Gitter in die untere Hälfte des Screens gezeichnet. Nachdem der Monitor optimal eingestellt ist, kehrt man durch das Drücken der Taste »RETURN« in das CLI zurück. Dieses Programm sollte nur vom CLI aus gestartet werden, da es beim Starten von der Workbench aus nicht beendet werden kann. Das gleiche gilt auch für einige weitere Demonstrationen, wie zum Beispiel für die »Dosedemo«. Das solche Programme nicht von der Workbench aus gestartet werden können, liegt daran, daß die DOS-Befehle, sowie »scanf« und »printf« nur auf ein CLI-Fenster reagieren. Das heißt, die Programme können zwar schon von der Workbench aus gestartet werden, aber sie führen die oben genannten Befehle nicht richtig aus.

1.6 Die Bibliotheken

Die Hardware des Amiga ist von einer Vielzahl von leistungsstarken Software-Modulen umgeben. Durch diesen modularen Aufbau bieten sich ungeahnte Möglichkeiten. Das System wird somit flexibler und leistungsstärker. Module können hinzugefügt oder, falls notwendig, verändert werden.

Einen Teil dieser Amiga-System-Software-Module bilden die Libraries, zu deutsch (Software-)Bibliotheken. Das Amiga-System enthält bisher 16 Module. Hier eine Übersicht:

| | |
|--------------|--|
| clist.lib | Enthält einige nützliche Routinen, die den Umgang und die Anwendung der Copper-Liste vereinfachen. |
| console.lib | Dieses Library enthält Programme für den Umgang mit der Tastatur, der sogenannten Console. |
| diskfont.lib | Das diskfont.lib ermöglicht die Verwendung der verschiedenen Schrifttypen, die sich auf der Workbench-Diskette befinden. |
| dos.lib | Durch dieses Library wird dem Amiga unter anderem der Zugriff auf die Diskette ermöglicht. Der Zugriff auf die Diskette ist dank dieses Libraries fast so einfach, wie von der Benutzerschnittstelle CLI aus. |
| exec.lib | Dieses Library bildet den System-Kern des Amiga. Dieser Kern entscheidet z.B. welche Tasks zum Laufen kommen (in der Computersprache bezeichnet man dies mit Scheduling) oder wieviel Speicherplatz für ein Programm bereitgestellt werden muß. |
| graphic.lib | Ohne Grafik geht heutzutage nichts mehr. Das graphic.lib ist ein sehr leistungsstarkes und umfangreiches Bibliotheksmodul, dessen Funktionen unter anderem durch den direkten Zugriff auf den Blitter und Copper phantastische Geschwindigkeiten in punkto Grafik sowie Animation ermöglichen. |

| | |
|---------------------|--|
| icon.lib | Hier sind verschiedene, durchaus nützliche Utilities für den Umgang mit den, von der Workbench her bekannten Icons enthalten. Es ist ebenfalls eines der wenigen Libraries, die sich auf der Workbench-Disk befinden. |
| info.lib | Dieses Library wird dazu verwendet um Information über Dateien, Datei-Verzeichnisse oder ganze Disketten zu bekommen. Es wird kaum verwendet und befindet sich auf der Workbench-Diskette. |
| intuition.lib | Das Intuition.lib ist eines der wichtigsten Libraries des Amiga. Ohne dieses Library wäre keine Bedienung mit der Maus oder die einfache Handhabung von Menüs denkbar. |
| janus.lib | Dies ist bisher das letzte Bibliotheks-Modul, das dem Amiga beigefügt wurde. Es befindet sich ebenfalls auf der Diskette und wird zur Steuerung der Side-Car Hardware benötigt. |
| layers.lib | In diesem Bibliotheks-Modul sind Routinen enthalten, die dem Anwender beispielsweise das Handling von überlappenden Display-Elementen erleichtern. |
| mathffp.lib | Mit diesem sogenannten FFP-Basic-Mathematik-Library können einfache mathematische Aufgaben, wie z.B. die Multiplikation oder Division, gelöst werden. |
| mathieeedoubbas.lib | Dies ist das erweiterte FFP-Basic-Mathematik-Library. Es befindet sich auf der Workbench-Diskette und enthält eine Vielzahl von mathematischen Funktionen, die Zahlen im IEEE-Standard mit doppelter Genauigkeit verarbeiten. |
| mathtrans.lib | Für schwierigere mathematische Aufgaben, wo Funktionen wie arcsin, arccos u.s.w Verwendung finden, enthält dieses Bibliotheks-Modul genügend Befehle. Da diese Funktionen nicht ständig verwendet werden, ist dieses Library auf der Workbench-Disk enthalten. |

| | |
|----------------|--|
| timer.lib | Wenn Sie zeitlich im Bilde sein wollen, bietet sich die Verwendung dieses Libraries an. Leider kann beim Amiga 1000 nur die Software-Uhr angesprochen werden. Bei den Versionen 500 und 2000 ist diese Uhr jedoch batteriegepuffert. |
| translator.lib | Das translator.lib hat die Aufgabe Sätze, die in englisch verfaßt sind, für die Sprachausgabe vorzubereiten. Es findet kaum Verwendung und ist deshalb auf der Systemdisk enthalten. |

Je nach Art des Programms, das der Programmierer entwickeln will, muß er selbständig entscheiden, welche Bibliotheks-Module er benötigt. Sicherlich werden Sie nun denken, je mehr Libraries verwendet werden, desto besser wird das Programm. Im Gegenteil! Für sehr gute Programme reichen schon 2 bis 3 Libraries aus.

Beim Umgang mit den Libraries müssen bestimmte Regeln eingehalten werden, damit die jeweiligen Funktionen ansprechbar sind. So hat es z.B. keinen Zweck, Funktionen eines Libraries aufzurufen, wenn das jeweilige Library nicht geöffnet wurde.

Bevor jedoch das jeweilige Library geöffnet wird, muß der »Basis« des Libraries ein Zeiger zugewiesen werden, hier am Beispiel des Intuition Library demonstriert, der von OpenLibrary zurückgegeben wird:

```
IntuitionBase = (struct IntuitionBase *)
    OpenLibrary("intuition.library", 0);
```

Anschließend enthält IntuitionBase die Einsprungsadresse der Intuition-Library. Enthält diese Variable den Wert »NULL«, war es nicht möglich, das Library zu öffnen.

Ist der Wert ungleich »NULL«, verlief alles normal und das Library konnte geöffnet werden.

Nachdem Sie ein Library geöffnet haben, muß es natürlich auch wieder geschlossen werden:

```
CloseLibrary(IntuitionBase);
```

schließt das jeweilige Library.

Hier zum besseren Verständnis nochmals ein Beispiel :

```
/* Öffnen und Schließen eines Bibliotheksmodules */
.
.
struct GfxBase *GfxBase; /* Zeiger für die Einsprungadresse */
. /* deklarieren */
.
main()
{
.
GfxBase = (struct GfxBase *)
OpenLibrary("graphics.library", 0); /* Library öffnen */
if (GfxBase == NULL)
{
printf("Öffnen des graphics.library nicht möglich !\n");
exit(FALSE);
}

/*
Hier das jeweilige Programm eintragen
*/

/* Zum Schluß Bibliotheks-Module schließen */

CloseLibrary(GfxBase);
}
```

IntuitionBase und GfxBase dürfen mit struct IntuitionBase *IntuitionBase; bzw. struct GfxBase *GfxBase; deklariert werden, da sie die Einsprungadressen der Intuition- und der Graphics-Library darstellen. Für alle anderen Libraries gilt folgende Deklaration:

Beispiel Diskfont-Library

```
.
.
ULONG DiskfontBase;
.
.
main()
{
.
DiskfontBase = OpenLibrary("diskfont.library",0);
if (DiskfontBase == NULL)
{
printf("Öffnen des diskfont.library nicht möglich !\n");
exit(FALSE);
}

/*
        Hier das jeweilige Programm eintragen
*/

/* Zum Schluß Bibliotheks-Module schließen */

CloseLibrary(DiskfontBase);
}
```

1.7 Die Devices

Neben den Libraries, die für den Programmierer Erleichterungen und für das System eine große Flexibilität darstellen, steht dem Programmierer weiteres großes Hilfsmittel zur Verfügung: die Devices.

Devices, zu deutsch Vorrichtungen, sind die Bindeglieder zwischen der (externen) Hardware und der Software des Amiga. Durch sie können Daten zur Hardware gesendet oder von ihr empfangen werden. Somit ist, z.B. durch das Verändern von Parametern der Trackdisk-Device, das Lesen von fremden Diskettenformaten, wie IBM oder Apple-Format möglich.

Der Amiga enthält 17 verschiedene Devices, die sich um die Vorrichtungen, wie Tastatur, der seriellen und parallelen Schnittstellen und einiges mehr kümmern. Nicht alle werden ständig benötigt, sondern befinden sich im »Devs«-Directory auf der Workbench-Disk.

Die Devices des Amiga im Überblick:

| | |
|-------------------|--|
| audio.device | Mit ihr wird der »Sound« des Amiga gesteuert. Je nach Belieben richtet sie die 4 Audio-Kanäle des Amiga ein, bestimmt die Amplitude des Tons und vieles mehr. |
| bootblock.device | Testet, ob es sich um eine Kickstart- oder um eine DOS-Diskette handelt. Bei den neuen Amigas ist diese Vorrichtung weggefallen, da bei ihnen keine Kickstartdiskette mehr erforderlich ist. |
| clipboard.device | Wird benötigt, um Daten zwischen zwei Anwendungen zu transferieren. Da dies nicht häufig vorkommt, befindet sich diese Device auf der Workbench-Disk. |
| console.device | Regelt die Ein- und Ausgabe des Systems über die Tastatur und den Bildschirm. |
| gameport.device | Gameport.device übernimmt die Steuerung der Ein- und Ausgabe über die GamePorts 1 und 2. |
| input.device | Diese Device regelt die gesamte Ein- und Ausgabe des Amiga. Es ist eine Kombination aus timer-, gameport- und keyboard.device. |
| inputevent.device | Inputevent.device erfaßt die Ereigniseingaben, wie z.B. Gadgets. |

| | |
|------------------|--|
| jdkisk.device | Dies ist die neuste Device des Amiga. Sie übernimmt die Steuerung der Harddisk des Amiga, die sich auf der IBM-PC kompatiblen Seite des Amiga2000 oder im SideCar befindet. Da sie sehr neu ist, befindet sie sich ebenfalls auf der Workbench-Disk. |
| keyboard.device | Hiermit wird der Zugriff auf die Tastatur des Amiga gesteuert. |
| keymap.device | Damit kann die Belegung der Tastatur verändert werden. |
| narrator.device | Narrator.device ist für die Steuerung der Sprachausgabe notwendig. Da sie nicht ständig benötigt wird, befindet sie sich auf der Workbench-Disk. |
| parallel.device | Hiermit kann der Parallelport gesteuert werden. Diese Device befindet sich ebenfalls auf der Workbench-disk. |
| printer.device | Diese Device dient zur Kommando-Steuerung des Druckers, um z.B. einen Wagnenlauf des Druckers zu bewirken. Printer.device befindet sich ebenfalls auf der Workbench-Disk. |
| prtbase.device | Prtbase.device übernimmt die Datendefinition der printer.device. |
| serial.device | Diese dient zur Deklaration des seriellen Ports des Amiga. Sie befindet sich ebenfalls auf der Workbench-Disk. |
| timer.device | Mittels Timer.device kann auf die Systemzeit zugegriffen werden. |
| trackdisk.device | Diese Device kontrolliert die Floppies des Amiga. Sie übernimmt Funktionen, wie das Lesen und Schreiben von Daten und einiges mehr. |

Um mit einer Devices arbeiten zu können, muß sie geöffnet werden. Dies geschieht mit

```
printerPort = CreatePort("printer.port",0);
```

wobei printerPort zurückgegeben wird.

Danach muß die Device geöffnet werden, in diesem Fall "printer.device":

```
fehler = OpenDevice("printer.device",0,&request,0);
```

Wenn Fehler gleich ungleich 0 ist, konnte die Device nicht geöffnet werden. `&request` ist der Pointer auf eine Structure der jeweiligen Device, die bestimmte »Routinen« wie z.B. das Drucken eines Screens enthalten oder auf die allgemeine Ein/Ausgabe-Structure von EXEC.

Nachdem die Device und der Port geöffnet sind, kann die benötigte Ein- und Ausgabe-Structure initialisiert werden.

Nach der Initialisierung wird die jeweilige »Funktion«, wie z.B. das Drucken eines Textes, mit

```
DoIO(&request);
```

gestartet.

`&request` ist der Pointer auf die Ein- und Ausgabe-Structure der jeweiligen »Funktion«.

Nachdem die Ein- und Ausgabe beendet ist, muß der Port und die Device wieder geschlossen werden. Dies kann mit

```
DeletePort(printerPort);
CloseDevice(&prefrequest);
```

erledigt werden.

```

1  /*****
2
3      Testbild
4      last update 26/05/87
5  von Frank Kremser und Joerg Koch
6      (c) Markt & Technik 1987
7
8  *****/
9
10 Dieses Programm erzeugt ein Testbild, mit dem der Monitor auf die idealen
11 Werte eingestellt werden kann.
12
13 *****/
14
15 #include <exec/types.h>          /* Include-Files einladen */
16 #include <intuition/intuition.h>
17
18 struct IntuitionBase *IntuitionBase;
19 struct GfxBase *GfxBase;
20
21 struct NewScreen ns =           /* Screen definieren */
22 {
23     0,
24     0,
25     640,
26     512,
27     4,
28     0,1,
29     HIRES:LACE,
30     CUSTOMSCREEN,
31     NULL,
```

```

32     NULL,
33     NULL,
34     NULL,
35     };
36
37 main()
38 {
39     LONG zahl;
40     char taste;
41     struct Screen *screen;
42
43     IntuitionBase = (struct IntuitionBase *) /* Intuition oeffnen */
44     OpenLibrary("intuition.library",0);
45     if(IntuitionBase == NULL) exit(FALSE);
46
47     GfxBase = (struct GfxBase *) /* Graphics oeffnen */
48     OpenLibrary("graphics.library",0);
49     if(GfxBase == NULL) exit(FALSE);
50
51     if((screen=(struct Screen*) /* Screen oeffnen */
52     OpenScreen(&ns)) == NULL) exit(FALSE);
53
54     SetRGB4(&screen->ViewPort,0,0,0,0); /* Farben setzen */
55     SetRGB4(&screen->ViewPort,1,15,15,15);
56     SetRGB4(&screen->ViewPort,2,15,0,0);
57     SetRGB4(&screen->ViewPort,3,0,15,0);
58     SetRGB4(&screen->ViewPort,4,0,0,15);
59
60     SetAPen(&screen->RastPort,1); /* 1. Farb-Quadrat */
61     SetDrMd(&screen->RastPort,JAM1);
62     RectFill(&screen->RastPort,0,0,639,511);
63
64     SetAPen(&screen->RastPort,2); /* 2. Farb-Quadrat */
65     RectFill(&screen->RastPort,0,0,211,256);
66
67     SetAPen(&screen->RastPort,3); /* 3. Farb-Quadrat */
68     RectFill(&screen->RastPort,214,0,427,256);
69
70     SetAPen(&screen->RastPort,4);
71     RectFill(&screen->RastPort,429,0,639,256); /* 4. Farb-Quadrat */
72
73     SetAPen(&screen->RastPort,0);
74     RectFill(&screen->RastPort,170,284,470,484); /* 5. Farb-Quadrat */
75
76     SetAPen(&screen->RastPort,1);
77
78     for(zahl=284; zahl<484; zahl = zahl + 20)
79     {
80         Move(&screen->RastPort, 170, zahl); /* Schaerfe - Gitter zeichnen */
81         Draw(&screen->RastPort, 470, zahl);
82     }
83
84     for(zahl=170; zahl<470; zahl = zahl + 20)
85     {
86         Move(&screen->RastPort, zahl, 284);
87         Draw(&screen->RastPort, zahl, 484);
88     }
89
90     scanf("%c", &taste); /* wenn RETURN, dann zurueck */
91
92     CloseScreen(screen); /* Screen und Libs schliessen */
93     CloseLibrary(IntuitionBase);
94     CloseLibrary(GfxBase);
95 }

```


Der Screen

Zunächst, bevor wir mit dem Umgang der Screens beginnen können, müssen wir festlegen was überhaupt ein Screen ist, was er benötigt und aus welchen »Bauteilen« er besteht.

Nun, was sind Screens? Man könnte Screens als die Grundlage aller Darstellungsformen von Intuition bezeichnen. Ein Screen ist also einer von vielen möglichen virtuellen Bildschirmen, auf dem verschiedene Fenster, sogenannte Windows, geöffnet werden können.

Diese Screens können beim AMIGA verschiedene Auflösungen und Farbtiefen annehmen. Horizontal gibt es die Möglichkeit einen Screen mit 640 oder 320 Pixels darzustellen. Dank des PAL-AMIGA ist die vertikale Auflösung, im Vergleich zu AMIGA's Anfangszeiten, etwas erhöht worden, sie beträgt nun 256 ohne und 512 Pixels mit Zeilensprungverfahren. Zeilensprungverfahren bedeutet, daß sich der Screen aus zwei zusammengesetzten Halbbildern zusammensetzt. Die Folge ist, daß die Bildfrequenz nun nur noch 25 Hz beträgt, weshalb die Darstellung leicht flackert. Es können auch mehrere Screens gleichzeitig in verschiedenen Auflösungen geöffnet werden. Merken sollte man sich nur: wenn einmal Interlace (Zeilensprungverfahren) eingeschaltet ist, so flackern auch die anderen geöffneten Screens im Hintergrund, auch dann, wenn sie keinen Interlace-Modus verwenden.

Der Anzahl der Screens ist keine bekannte softwaremäßige Grenze gesetzt. Das einzige Hemmnis, das wir gefunden haben, ist der begrenzte Speicherplatz, denn die Video-Hardware des Amiga kann nur die untersten 512 KByte für den Aufbau von Screens ansprechen. Dieser Speicherbereich wird auch CHIP-Memory genannt, da die Chips nur auf diesen Bereich zugreifen können. Der darüberliegende Speicherbereich heißt FAST-Memory, da die CPU beim Zugriff auf diesen Bereich nicht von den Custom-Chips gebremst wird.

Viel Speicherplatz wird auch dann benötigt, wenn viele Farben in einem Screen und somit viele Bit-Planes dargestellt werden sollen. Der Amiga bietet in punkto Farben eine sehr breite Palette an. Drei »Farbregler«, je einen für rot, grün und blau, ermöglichen 16 mal 16 mal 16 gleich 4096 Farben. Diese maximale Anzahl von Farben ist aber leider nur in einem besonderen Modus, dem HAM-Modus (Hold-And-Modify, zu deutsch »halte und verändere«) bei einer Screenauflösung von 320 mal 256 Pixels gleichzeitig darstellbar. Hierbei wird mit einer Bit-Plane-Tiefe von 6 gearbeitet. Tiefe 5 und 6 werden dazu benötigt, um die Tiefen von 1 bis 4 zu verändern. Im HAM-Modus benötigt der Amiga rund 48 KByte für den Screen. Dies berechnet sich aus der Auflösung 320 mal 256 Pixels mal der Bit-Plane-Tiefe 6, geteilt durch 8, da 8 Bit ein Byte sind.

Leider kann auch nicht in jedem Darstellungsmodus die maximale Anzahl von 6 Bitplanes, bzw. Bit-Tiefen, dargestellt werden. Dazu siehe Tabelle Nr. 1.

| Auflösung | max. Bit-Tiefe | Farben | benöt. Speicherplatz |
|-------------------|----------------|--------|----------------------|
| 320 x 256 | 5 | 32 | 51 200 Byte |
| 320 x 512 | 5 | 32 | 102 400 Byte |
| 640 x 256 | 4 | 16 | 81 920 Byte |
| 640 x 512 | 4 | 16 | 163 840 Byte |
| Besondere Modi: | | | |
| Extra-Halfbright: | | | |
| 320 x 256 | 6 | 64 | 61 440 Byte |
| 320 x 512 | 6 | 64 | 122 880 Byte |
| HAM:320 x 256 | 6 | 4096 | 61 440 Byte |

Tabelle Nr. 1

Sicherlich wird nun bei einigen Lesern die Frage auftauchen, was ein Screen, neben viel Speicherplatz, noch so alles benötigt. Nun da wäre zunächst eine Liste, die die Verwaltung des Speicherplatzes und somit die Definition des Screens übernimmt. Diese Liste wird als ViewPort-Liste bezeichnet. Von ihr können weitere Informationen über die Farbtabelle der verwendeten Farben, sowie über die Größe und Lage der Bit-Maps und der Copper-Liste, die zur Steuerung der Video-Hardware benötigt wird, abgeleitet werden. Eng verwandt mit der ViewPort-Liste ist die RastPort-Liste. Diese Liste definiert einen Ausschnitt des Screens, somit also ein Window. Es kann natürlich auch vorkommen, daß dieser Ausschnitt den ganzen Screen umfaßt.

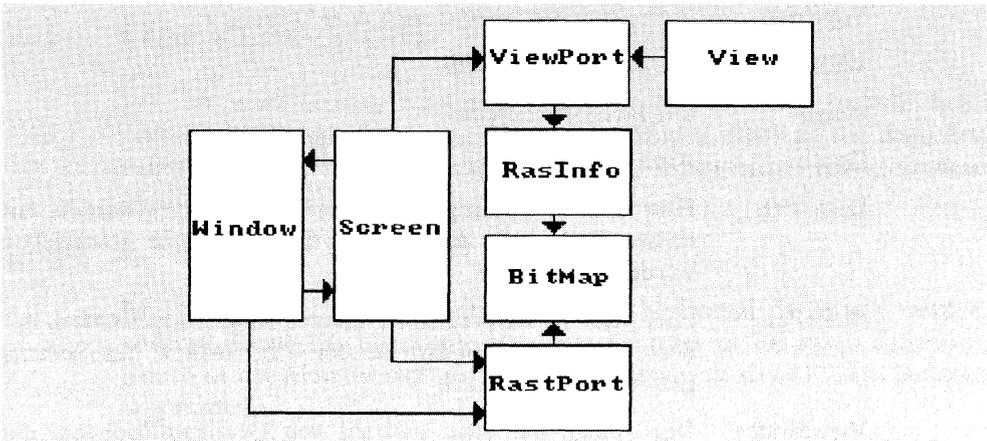


Bild 2.1 Zeigt den Zusammenhang zwischen Window, Screen, Rast- und ViewPort

Damit dem Anwender der Umgang mit Rast- und ViewPort, sowie dem Bereitstellen von Speicherplatz nicht so schwer fällt, stellt Intuition nützliche Hilfsmittel zur Verfügung. Die Screens, die mit diesen Hilfsmitteln erstellt werden, bezeichnet man als Custom-Screens. Bevor jedoch das Bemalen eines solchen Screens losgehen kann, muß der Anwender Intuition mit einer NewScreen-Structure die gewünschten Daten des Screens mitteilen. Aus diesen Daten und ein paar anderen Befehlen erstellt dann Intuition die benötigten Listen, ohne daß der Programmierer sich noch darum zu kümmern braucht.

Die NewScreen-Structure hat folgenden Aufbau:

```

struct NewScreen
{
    SHORT LeftEdge, TopEdge;
    SHORT Width, Height, Depth;
    UBYTE DetailPen, BlockPen;
    USHORT ViewModes;
    USHORT Type;
    struct TextAttr *Fonts;
    UBYTE *DefaultTitle;
    struct Gadget *Gadgets;
    struct BitMap *CustomBitMap;
};
  
```

Da die Structure-Namen von Commodore festgelegt sind und in ihrer Form fest in der Sprache C auf dem AMIGA implementiert sind, müssen wir die englischen Namen übernehmen. Die einzelnen Variablen haben folgende Bedeutung:

LeftEdge X-Position des Screens (noch nicht verwendbar).

| | |
|-----------|--|
| TopEdge | Y-Position des Screens nach dem Öffnen. |
| Width | Die Breite des Screens. |
| Height | Die Höhe des Screens. |
| Depth | Anzahl der Bit-Planes (Farbanzahl = 2^{Depth}). |
| DetailPen | Hier muß die Farbbregister-Nummer eingetragen werden, mit dessen Farbe z.B. der Text in der Titel-Zeile geschrieben werden soll. |
| BlockPen | Hier muß die Farbbregister-Nummer eingetragen werden, mit dessen Farbe beispielsweise der Titel-Balken des Screens gezeichnet werden soll. |
| ViewModes | Der Amiga hat eine Vielzahl von Grafikauflösungen, die durch »ViewModes« bestimmt werden können: |

DUALPF Ist dieses Flag gesetzt, so können zwei sogenannte »Play-Fields« verwendet werden.

HAM Wenn dieses Flag gesetzt wird, befindet sich der Amiga im »Hold-And-Modify – Modus«, kurz »HAM«.

HIRES Ist dieses Flag gesetzt, so ist die maximale horizontale Auflösung 640 Pixels, andernfalls 320 Pixels.

INTERLACE Durch das »Interlace-Flag« wird der Zeilensprung aktiviert. Somit sind vertikal max. 512 Pixels möglich. Ist dieses Flag nicht gesetzt, so beträgt die max. Auflösung vertikal nur 256 Pixels.

SPRITES Wenn Hardware-Sprites verwendet werden, so muß dieses Flag gesetzt sein.

Beispiel: Soll die max. Auflösung 640 mal 512 Pixels betragen, so muß in »View-Modes« folgendes stehen:

HIRES|INTERLACE

Soll jedoch die geringe Auflösung von 320 mal 256 eingeschaltet sein, so kann »View – Modes« auf »NULL« gesetzt werden.

Weitere Informationen folgen auf den nächsten Seiten.

2.1 Die View-Modi

Wie im einführenden Kapitel »Screens« schon erwähnt, besteht beim AMIGA die Möglichkeit, verschiedene ViewModes, damit ist das Aussehen der Screens gemeint, zu wählen. Der Amiga kennt bisher 8 verschiedene Modi.

HIRES

Ist HIRES gesetzt, so können 640 Punkte horizontal dargestellt werden, andernfalls nur 320. Die Video-Hardware kann bis zu 700 Pixels darstellen, jedoch ist der Monitor nicht für größere Auflösungen als 640 Pixels horizontal geeignet.

LACE

Mit LACE wird die Video-Hardware des Amiga auf das Zeilensprungverfahren eingestellt. Dieses Verfahren wird auch beim Farbfernseher verwendet. Dabei wird jedes Bild aus zwei Halbbildern zusammengesetzt. Im ersten Durchgang »schreibt« die Video-Hardware des Amiga alle geraden, im zweiten alle ungeraden Zeilen. In der Regel benötigt ein kompletter Bildaufbau ohne Zeilensprungverfahren 1/60 Sekunde. Da aber nun 2 Bilder dargestellt werden müssen, verdoppelt sich die Zeit. Sie beträgt nun für den kompletten Bildaufbau 1/30 Sekunde. Dafür können mit LACE nun 512, anstatt nur 256 Zeilen dargestellt werden. Da der Monitor mit einer Bildfrequenz (Zeit für den Bildaufbau) von 50 Hz, das ist 1/50 Sekunde, arbeitet, fängt beim Einschalten des LACE-Modus das darzustellende Bild des AMIGA, leicht zu flackern an.

HAM

Dies ist ein Spezialmodus des Amiga. Er erlaubt die Darstellung aller 4096 Farben zur selben Zeit, bei einer Auflösung von 320 x 256 Pixels. Normal ist es so, daß die Anzahl der BitMaps die maximale Anzahl der Farbregister und so die Anzahl der Farben bestimmt. Da maximal nur 6 BitMaps möglich sind, können rein theoretisch nur $2^6 = 64$ verschiedene Farben dargestellt werden. Nun, wie kommt man in HAM zu 4096 Farben ?

Der Name HAM verrät schon den ganzen Trick. HAM bedeutet »Hold And Modify«, zu deutsch »Halte und verändere«. Im HAM-Modus werden alle 6 Bit-Maps verwendet. Die Bit-Maps 5 und 6 haben in diesem Modus eine besondere Bedeutung, sie werden dazu benutzt, die Bit-Kombinationen der Bit-Maps 1 bis 4 zu verändern. Ein HAM-Modus muß somit immer 5 oder 6 Bit-Maps haben.

Das Ganze läuft nun so ab, daß jeweils die Bit-Kombination der Bit-Maps 5 und 6 ermittelt werden. Es gibt also insgesamt $2^2 = 4$ Kombinationsmöglichkeiten für die Bit-Maps 5 und 6:

Ist die Kombination gleich 00, wird die normale Farb-Selektion durchgeführt. Das heißt, die Bit-Kombination der Planes 1 – 4 wird dazu verwendet, eines der Farb-Register 0 bis 15 auszuwählen. Ist die Kombination gleich 01, wird die Bit-Kombination der Bit-Maps 1 bis 4 dazu verwendet, den roten Farbwert des zuletzt angewählten Farbreisters zu verändern. Ist die Kombination gleich 10, wird die Bit-Kombination der Bit-Maps 1 bis 4 dazu verwendet, den grünen Farbwert des zuletzt angewählten Farbreisters zu verändern. Ist die Kombination gleich 11, wird die Bit-Kombination der Bit-Maps 1 bis 4 dazu verwendet, den blauen Farbwert des zuletzt angewählten Farbreisters zu verändern.

Werden nur 5 Bit-Maps verwendet ist der Wert des 6. Bit-Maps automatisch 0, was bedeutet, daß entweder ein Farbreister mittels der Bitkombination der Planes 1 bis 4 ausgewählt wird, oder, wenn der Wert in der fünften Bit-Plane gleich 1 ist, der Rot-Anteil des zuletzt angewählten Farbreisters modifiziert wird.

GENLOCK_VIDEO

Dieses Flag muß gesetzt sein, wenn mit dem Genlock Video Interface experimentiert wird. Dabei werden die Signale zur Darstellung des Bildschirms mit einer fremden Videoquelle synchronisiert. Die Hintergrundfarbe wird gegen die fremde Videoquelle »ausgetauscht«. Während »Miami Vice« im Hintergrund läuft, kann im Vordergrund mit DeluxePaint gearbeitet werden.

EXTRA_HALFBRIGHT

Dieses Bit schaltet, wenn es gesetzt ist, den Amiga in einen Spezial-Modus, der bei einer Auflösung von 320 x 256 Pixels und 320 x 512 Pixels eine Bit-Map-Anzahl von 6 und somit die Darstellung von 64 verschiedenen Farben erlaubt.

SPRITES

Dieses Flag muß gesetzt werden, wenn Hardware-Sprites verwendet werden sollen.

VP_HIDE

Ist dieses Flag gesetzt, wird zwar ein Screen erzeugt, aber nicht dargestellt.

2.2 Die NewScreen-Structure

Intuition stellt eine Reihe von nützlichen Hilfsmitteln bereit, die das Erstellen und Anwenden von Custom-Screens erheblich vereinfachen. Damit Intuition einen Custom-Screen bilden kann, muß der Anwender zunächst verschiedene Daten, die den Screenaufbau beschreiben, übergeben. Diese Daten sind einer NewScreen Struktur enthalten.

Die NewScreen Struktur:

```
struct NewScreen
{
    SHORT  LeftEdge, TopEdge;
    SHORT  Width, Height, Depth;
    UBYTE  DetailPen, BlockPen;
    USHORT ViewModes;
    USHORT Type;
    struct TextAttr *Fonts;
    UBYTE  *DefaultTitle;
    struct Gadget *Gadgets;
    struct BitMap *CustomBitMap;
};
```

Die einzelnen Variablen haben folgende Bedeutungen:

| | |
|----------|--|
| LeftEdge | X-Positon des Screens (derzeit keine Wirkung). |
| TopEdge | Y-Position des Screens nach dem Öffnen. |
| Width | Die Breite des Screens. Sie ist abhängig von dem verwendeten ViewModi: ViewMode = nicht HIRES = max. 320 Pixels, ViewMode = HIRES = max. 640 Pixels. Die Video-Hardware des AMIGA kann sogar eine Auflösung bis zu 700 Pixels horizontal darstellen. Leider ist aber der Monitor des AMIGA für eine solche Auflösung nicht mehr geeignet. |
| Height | Die Höhe des Screens. Auch Sie ist abhängig von dem verwendeten ViewModi: ViewMode = nicht LACE = max. 256 Pixels, ViewMode = LACE = max. 512 Pixels. |

Depth Die Tiefe des Screens. Damit ist die Anzahl der BitMaps und die damit verbundene Anzahl der Farben gemeint. Maximal sind 6 BitMaps möglich. Aber nicht in allen Darstellungsarten sind sie verwendbar. Die maximale Anzahl der Bit-Maps, die noch verwendbar sind, ist abhängig von dem jeweiligen View-Mode und somit von der verwendeten Auflösung:

| Auflösung: | ViewMode: | max. BitMaps | max. Farben |
|------------|-----------------------|--------------|-------------|
| 320 x 256 | - | 5 | 32 |
| 320 x 256 | EXTRA_HALFBRIGTH | 6 | 64 |
| 320 x 256 | HAM | 6 | 4096 |
| 320 x 512 | LACE | 5 | 32 |
| 320 x 512 | LACE EXTRA_HALFBRIGTH | 6 | 64 |
| 640 x 256 | HIRES | 4 | 16 |
| 640 x 512 | HIRES LACE | 4 | 16 |

DetailPen Hier muß die Farbreister-Nummer eingetragen werden, die z.B. für den Text in der Titelzeile verwendet werden soll.

BlockPen Hier muß die Farbreister-Nummer eingetragen werden, die beispielsweise für den Titel-Balken des Screens verwendet werden soll.

ViewModes Der AMIGA hat eine Vielzahl von Grafikaufösungen, die durch »ViewModes« bestimmt werden können, die schon im Zusammenhang mit der Bit-Tiefe erwähnt wurden.

Type Hier wird im Normalfall CUSTOMSCREEN eingetragen. Theoretisch kann hier auch WBENCHSCREEN verwendet werden, wodurch eine eigene Workbench geöffnet werden kann, was aber keinen allzu großen Zweck hat. Genaueres siehe unter der Screen-Structure.

Fonts Falls ein besonderer Font (Zeichensatz) Verwendung finden soll, muß hier der Zeiger auf seine Structure eingetragen werden. Soll jedoch der momentane Intuition-Font verwendet werden, trägt man an dieser Stelle NULL ein.

Soll ein anderer Zeichensatz verwendet werden, bietet sich folgende Lösung:

```
struct TextAttr Font =
{
    "emerald.font",  Zeichensatzname
    20,              Zeichensatzhöhe
    FS_NORMAL,      Darstellungsart
    FPF_DISKFONT    Fontauf Diskette}
```

Eingetragen wir dann an dieser Stelle in die NewScreen Structure »&Font«.

- Title** Hier wird der Titel für den Screen-Balken eingetragen. Falls kein Titel benötigt wird, kann einfach »NULL« eingetragen werden.
- Gadgets** Hier wird in der Regel immer »NULL« eingetragen, da das Setzen des Gadgetpointers bei Verwendung von Gadgets von Intuition selbst übernommen wird.
- CustomBitMap** Dieser Zeiger muß auf eine selbst erstellte BitMap-Struktur zeigen, wenn CUSTOMBITMAP verwendet wird, ansonsten ist der Wert dieser Variable »NULL«.

Übergeben wird Intuition diese NewScreen-Structure mit OpenScreen(). Intuition fügt diese Parameter in eine größere Struktur, die Screen-Structure, ein. Beim Öffnen des Screens übergibt Intuition den Pointer auf diese größere Struktur automatisch an das Programm zurück. Er ist sehr wichtig für weitere Anwendungen, wie z.B. für das Zeichnen auf den Screen.

Zur Verdeutlichung ein Beispiel:

```
/* definieren der größeren Structure */
struct Screen *screen;

/* nun Definition der New-Screen-Structure */
struct NewScreen newscreen =
{
    0,          /* x - Parameter */
    0,          /* y - Parameter */
    320,
    256,       /* Auflösung   */
    6,         /* Tiefe       */
    0,
    1,         /* Zeichenstifte */
    HAM,       /* ViewModi    */
    CUSTOMSCREEN, /* Type       */
    NULL,      /* Font        */
}
```

```
    "HAM-DEMO",    /* Screen-Titel */
    NULL,         /* Gadget-Pointer*/
    NULL         /* BitMap-Pointer*/
}
main()
{
    /* Öffnen des Screens mit der NewScreen-Structure und Übernehmen des
       Pointers der größeren Screen-Structure */
    screen = (struct Screen *)OpenScreen(&newscreen);

    /* Setzen der Farbe mit Hilfe des Pointers auf die größere Structure, der
       Screen-Struktur => &screen. Dabei wird der ViewPort von der Screen-
       Struktur abgeleitet.
       &screen->ViewPort                                     */
    SetRGB4(&screen->ViewPort, 0, 15, 15, 15);
}
```

2.3 Die Screen-Structure

Wie Sie im vorhergehenden Abschnitt gelesen haben, ist die New-Screen Structure nur eine Art »Notiz-Zettel«, die Intuition nur das Nötigste für den Screenaufbau mitteilt. Dieser »Notiz-Zettel« wird nach dem Öffnen des Screens nicht mehr benötigt, da eine erweiterte Structure zurückgegeben wird.

Auf die erweiterte, größere Structure, die Screen-Structure wollen wir nun etwas näher eingehen:

Die Screen-Structure:

```
struct Screen
{
    struct Screen *NextScreen;
    struct Window *FirstWindow;
    SHORT LeftEdge, TopEdge, Width, Height;
    SHORT MouseY, MouseX;
    USHORT Flags;
    UBYTE Title;
    UBYTE DefaultTitle;
    BYTE BarHeight, BarVBorder, BarHBorder,
        MenuVBorder, MenuHBorder;
    BYTE WBotTop, WBotLeft, WBotRight, WBotBottom;
    struct TextAttr *Font;
    struct ViewPort ViewPort;
    struct RastPort RastPort;
    struct BitMap BitMap;
    struct Layer_Info LayerInfo;
    struct Gadget *FirstGadget;
    UBYTE DetailPen, BlockPen;
    USHORT SaveColor0;
    struct Layer *BarLayer;
    UBYTE *ExtData;
    UBYTE *UserData;
};
```

Im Vergleich zur NewScreen-Structure sind einige Daten, wie ViewPort und RastPort hinzugekommen. Dies sind wichtige Pointer, die z.B. zum Setzen der Farbe oder zum Zeichnen benötigt werden.

Im einzelnen bedeuten die Variablen folgendes:

| | |
|-------------------------------------|--|
| NextScreen | Dies ist ein Pointer auf den nächsten Screen. |
| FirstWindow | Dies ist ein Pointer auf das erste verwendete Window in diesem Screen. |
| LeftEdge, TopEdge, Width, Height | Screendimensionen, siehe NewScreen-Struktur |

| | |
|--|--|
| MouseY, MouseX | X- und Y-Koordinaten der Maus, relativ gesehen zur oberen linken Ecke; |
| Flags | Diese Flags werden von Intuition gesetzt. Sie beschreiben die unterschiedlichen Screens die unter Intuition möglich sind: SCREENTYPE = alle Definitionen sind möglich. Definitionen: WBENCHSCRN = Die Workbench. CUSTOMSCREEN = Benutzer-Screen. SHOWTITLE = Ist dieses Flag gesetzt, so wird die Titelleiste angezeigt. Dieses Flag kann mit ShowTitle() verändert werden. BEEPING = Dieses Flag wird gesetzt, wenn der Screen aufblinkt (beept), siehe DisplayBeep(). CUSTOMBITMAP = Wenn ein eigenes BitMap verwendet werden soll, muß dieses Flag gesetzt sein. |
| Title | Screen Titel der Titelleiste. |
| DefaultTitle | WindowTitel für Windows ohne Screen-Titel. |
| BarHeight, BarVBorder, BarHBorder, MenuVBorder, MenuHBorder | Dimensionen der Titel- und Menüleiste |
| WBorTop, WBorBottom, WBorRight, WBorLeft | Dimension der Titelleiste aller Windows in diesem Screen. |
| Font | Screen-Font. Siehe NewScreen-Struktur. |

| | |
|---------------------|---|
| ViewPort | Der Pointer auf den ViewPort des Screens. Der ViewPort enthält weitere X- und Y- Dimensionen des Screens sowie Pointer zu einer Liste, die den Copper und somit die Videodarstellung steuert. |
| RastPort | Pointer auf den RastPort des Screens. Der RastPort enthält eine Liste über die verwendeten Zeichenstifte, Zeichen-Modi u.s.w. |
| BitMap | Pointer auf das vom Benutzer definierte BitMap. Siehe NewScreen-Struktur. |
| LayerInfo | Layer Information. |
| FirstGadget | Pointer auf das erste verwendete Gadget. Siehe NewScreen-Struktur. |
| DetailPen, BlockPen | Zeichenstifte. Siehe NewScreen-Struktur. |
| SaveColor0 | Diese Variable enthält den Wert für die Farbe, die bei DisplayBeep verwendet wird. Sie wird von Intuition gesetzt. |
| BarLayer | Pointer für die Darstellung der Titel-leiste. |
| ExtData, UserData | Hier kann der Benutzer eigene Pointer auf seine eigenen Daten eintragen. |

Nicht alle Variablen und Pointer sind von großer Bedeutung. Merken sollten Sie sich jedoch, daß sie von dieser Structure, der Screen-Structure, sowohl an den RastPort- als auch an den ViewPort-Pointer gelangen können.

An den Pointer dieser Screen-Structure gelangen Sie, wie schon beschrieben, wenn Sie mit `OpenScreen()` die NewScreen-Structure an Intuition übergeben und somit einen Screen öffnen.

Möchten Sie nun nachträglich Veränderungen vornehmen, z.B. den Titel ändern, so können Sie das folgendermaßen:

```
&screen->Title = "Markt & Technik";
```

oder wenn Sie den RastPort benötigen, können Sie so darauf zugreifen:

```
&screen->RastPort;
```

Sie verwenden also jeweils den Pointer des Screens, in dem Fall also `&screen`, mit einem »Pfeil« auf den jeweiligen Eintrag, den Sie benötigen.

2.4 Die Screen-Befehle

2.4.1 CloseScreen

| | |
|------------|--|
| Syntax: | CloseScreen(Screen); |
| Funktion: | Ein geöffneter Screen wird geschlossen. |
| Parameter: | Screen -> Zeiger auf die Screen-Structure des Screens, der geschlossen werden soll. |
| Ergebnis: | Kein Ergebnis. |
| Datentyp: | struct Screen *Screen; |
| Sonstiges: | Wenn ein Screen mit OpenScreen(&Screen) geöffnet wurde, kann er mit CloseScreen(&Screen) geschlossen werden. |
| Referenz: | Siehe Anwendungsbeispiele HAM-, Extrahalfbright-, Screen-Demo. Siehe auch CloseScreen. |

2.4.2 CloseWorkbench

| | |
|------------|--|
| Syntax: | erfolg = CloseWorkbench(); |
| Funktion: | Diese Routine schließt den Workbench-Screen. Wenn die Workbench geöffnet ist, wird getestet, ob verschiedene Windows auf ihr geöffnet sind. Trifft dies zu, so bleibt die Workbench geöffnet und »erfolg« nimmt den Wert »FALSE« an. Falls keine Windows auf der Workbench geöffnet sind, wird der Workbench-Screen geschlossen und »erfolg« nimmt den Wert »TRUE« an. |
| Parameter: | Keine Parameter. |
| Ergebnis: | erfolg -> ist TRUE, wenn die WorkBench geschlossen werden konnte, ansonsten ist »erfolg« FALSE. |
| Datentyp: | bool erfolg; |
| Sonstiges: | Dieser Befehl kann angewendet werden, um Speicherplatz zu gewinnen. |
| Referenz: | Siehe Screen-Demo. |

2.4.3 DisplayBeep

- Syntax:** DisplayBeep(Screen);
- Funktion:** DisplayBeep(&Screen) läßt den Screen kurz in einer anderen Hindergrundfarbe aufblinken.
- Parameter:** Screen -> ist der Pointer auf die jeweilige Screenstruktur. Wird der Parameter mit »NULL« angegeben, so blinken alle geöffneten Intuition-Screens auf.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct Screen *Screen;
- Sonstiges:** Wenn der Programmanwender gewarnt werden soll. Ein Beispiel hierfür finden wir beim BASIC-Interpreter des AMIGA. Tritt ein Syntax-Error auf, so wird die DisplayBeep-Funktion aufgerufen.
- Referenz:** Siehe Screen-Demonstration.

2.4.4 MakeScreen

- Syntax:** MakeScreen(Screen);
- Funktion:** Richtet den ViewPort eines Intuition CustomScreens ein.
- Parameter:** Screen -> ist der Pointer auf die jeweilige Screen-Structure.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct Screen *Screen;
- Sonstiges:** Ist ein Screen gelöscht und es besteht der jeweilige Pointer zur Screen-Struktur noch, so kann mit MakeScreen(&Screen) der Screen wieder »zurückgeholt« werden.
- Nach dem Aufruf von MakeScreen(&Screen) kann mit Hilfe der Funktion RethinkDisplay() der neue ViewPort des Screens in die »Intuition Wiedergabe« aufgenommen werden. Gleichzeitig werden durch RethinkDisplay() alle anderen ViewPorts »aufgefrischt«.
- Referenz:** Siehe auch RethinkDisplay

2.4.5 MoveScreen

| | |
|------------|---|
| Syntax: | MoveScreen(Screen,dx,dy); |
| Funktion: | MoveScreen bewegt (scrollt) den angegebenen Screen in der Vertikalen. |
| Parameter: | Screen -> Zeiger auf die Screen-Structure des Screens, der bewegt werden soll. dx,dy -> Anzahl der Punkte, um die der Screen bewegt werden soll. Es sind auch negative Werte zulässig. |
| Ergebnis: | Kein Ergebnis. |
| Datentyp: | struct Screen *Screen; int dx, dy; |
| Sonstiges: | Derzeit kann noch nicht in der Horizontalen bewegt werden. Aus diesem Grund hat »dx« noch keine Bedeutung. Dieser Befehl hat die gleiche Wirkung, wie das »Herunterziehen« eines Screens mit der Maus. |
| Referenz: | Siehe auch Screen-Demo. |

2.4.6 OpenScreen

| | |
|------------|--|
| Syntax: | Screen = OpenScreen(NewScreen); |
| Funktion: | Öffnen eines Screens mit den Parametern, die in der NewScreen-Struktur festgelegt sind. |
| Parameter: | NewScreen -> Zeiger auf die NewScreen-Structure, die die Daten für den Screen enthält. |
| Ergebnis: | Screen -> Zeiger auf die Screen-Structure des geöffneten Screens. |
| Datentyp: | struct NewScreen *NewScreen; struct Screen *Screen; |
| Sonstiges: | Weitere Erklärungen finden Sie unter Kapitel 2, 2.1, 2.2 und 2.3. |

Referenz: Siehe auch ShowTitle
Screen-Demo
HAM-Demo
Extrahalfbrighth-Demo
Kapitel 2 DerScreen

2.4.7 OpenWorkBench

Syntax: `erfolg = OpenWorkbench();`

Funktion: `OpenWorkbench()` versucht die Workbench wieder zu öffnen.

Parameter: Keine Parameter.

Ergebnis: `erfolg` -> ist TRUE, wenn sie geöffnet werden konnte, ansonsten ist »erfolg« FALSE.

Datentyp: `bool` `erfolg`;

Sonstiges: Wenn mit `CloseWorkbench()` die Workbench geschlossen worden ist, kann sie mit `OpenWorkbench` wieder geöffnet werden.

Referenz: Siehe auch `CloseWorkbench()`.

2.4.8 RemakeDisplay

Syntax: `RemakeDisplay()`

Funktion: Diese Routine frischt alle augenblicklichen ViewPorts der Intuition-Screens durch Aufrufen der Funktion `MakeScreen(&Screen)` auf. Danach wird `RethinkDisplay` aufgerufen, welches die Relationen des Screens und die Darstellungsliste des Coppers auffrischt.

Parameter: Keine Parameter.

Ergebnis: Kein Ergebnis.

Datentyp: Keine Variablen.

Sonstiges: Diese Routine kann einige Millisekunden benötigen, bevor mit dem weiteren Ablauf des Programms vorgefahren wird.

Referenz: Siehe auch `RethinkDisplay`.

2.4.9 RethinkDisplay

| | |
|------------|---|
| Syntax: | RethinkDisplay(); |
| Funktion: | Zum Auffrischen der Relationen des Screens, sowie der Darstellungsliste des Coppers. |
| Parameter: | Keine Parameter. |
| Ergebnis: | Kein Ergebnis. |
| Datentyp: | Keine Variablen. |
| Sonstiges: | Diese Routine benötigt einige Millisekunden, bevor mit dem weiteren Programmablauf fortgefahren wird. |
| Referenz: | Siehe auch RemakeDisplay. |

2.4.10 ScreenToBack

| | |
|------------|---|
| Syntax: | ScreenToBack(Screen); |
| Funktion: | Der Screen wird in den Hintergrund gebracht und von anderen geöffneten Screens überdeckt. |
| Parameter: | Screen -> Zeiger auf die Screen-Structure des Screens, der in den Hintergrund gebracht werden soll. |
| Ergebnis: | Kein Ergebnis. |
| Datentyp: | struct Screen *Screen; |
| Sonstiges: | Wenn mit ScreenToFront der Screen in den Vordergrund gebracht wurde, kann er mit ScreenToBack(&Screen) wieder in den Hintergrund gebracht werden. |
| Referenz: | Siehe auch ScreenToBack WBenchToBack WBenchToFront |

2.4.11 ScreenToFront

| | |
|-----------|---|
| Syntax: | ScreenToFront(Screen); |
| Funktion: | Bringt den jeweiligen Screen in den Vordergrund, andere geöffnete Screens werden überlagert. Parameter: Screen -> Zeiger auf die Screen-Structure des Screens, der in den Vordergrund gebracht werden soll. |
| Ergebnis: | Kein Ergebnis. |

| | |
|------------|---|
| Datentyp: | struct Screen *Screen; |
| Sonstiges: | Wenn ein Screen mit ScreenToBack(&Screen) in den Hintergrund gebracht wurde, kann er mit ScreenToFront(&Screen) nach vorne gebracht werden. |
| Referenz: | Siehe auch ScreenToBack WBenchToBack WBenchToFront |

2.4.12 SetRGB4

| | |
|------------|---|
| Syntax: | SetRGB4(ViewPort,reg,rot,grün,blau) |
| Funktion: | Setzen eines Farbregisters des angegebenen Screens mit einer beliebigen Farbe. |
| Parameter: | ViewPort -> Zeiger auf den ViewPort des jeweiligen Screens. reg -> Nummer des Farbregisters rot -> Rot-Anteil. 16 Stufen möglich. grün -> Grün-Anteil. 16 Stufen möglich. blau -> Blau-Anteil. 16 Stufen möglich. |
| Ergebnis: | Kein Ergebnis. |
| Datentyp: | struct ViewPort *ViewPort; int reg, rot, grün, blau; |
| Sonstiges: | Der ViewPort-Pointer muß von dem jeweiligen Screen, wo die Farbe verändert werden soll, abgeleitet sein. Dies geschieht wie folgt: &Screen -> ViewPort. |
| Referenz: | Siehe Screen-Demonstrationen |

2.4.13 ShowTitle

| | | |
|------------|--|--|
| Syntax: | ShowTitle(Screen,mode); | |
| Funktion: | Durch diese Funktion kann bestimmt werden, ob die Titelleiste im Vordergrund oder im Hintergrund von »Backdrop«-Windows angezeigt werden soll. | |
| Parameter: | Screen | -> ist ein Zeiger auf die jeweilige Screen-Structure. |
| | Mode | -> muß TRUE gesetzt werden, wenn die Titelleiste vor Backdrop-Windows erscheinen soll. Ansonsten muß »mode« gleich FALSE sein. |
| Ergebnis: | Kein Ergebnis. | |
| Datentyp: | struct Screen *Screen; bool Mode; | |
| Sonstiges: | Nachdem ein Screen mit OpenScreen(&Screen) geöffnet wurde, wird ShowTitle automatisch auf »TRUE« gesetzt. Dies kann mit ShowTitle(&Screen, mode) verändert werden. | |
| Referenz: | OpenScreen. | |

2.4.14 WBenchToBack

| | | |
|------------|--|--|
| Syntax: | erfolg = WBenchToBack(); | |
| Funktion: | Die Workbench wird nach hinten gebracht und wenn andere Screens geöffnet sind, von diesen verdeckt. | |
| Parameter: | Keine Parameter. | |
| Ergebnis: | erfolg | -> ist TRUE, wenn die Workbench in den Hintergrund gebracht werden konnte. |
| Datentyp: | bool erfolg; | |
| Sonstiges: | Wenn die Workbench andere geöffnete Screens überdeckt, können diese durch WBenchToBack() nach vorne gebracht werden. | |
| Referenz: | Siehe auch ScreenToFront ScreenToBack WBenchToFront | |


```
39 struct NewScreen HighScreen =
40 {
41     0,
42     64,
43     640,
44     512,
45     2,
46     0,1,
47     HIRES:LACE,
48     CUSTOMSCREEN,
49     NULL,
50     "Screen mit 640 x 512 Pixels",
51     NULL,
52     NULL,
53 };
54
55 main()
56 {
57     struct Screen *LScreen;
58     struct Screen *HScreen;
59
60     LONG zahl;
61
62     IntuitionBase = (struct IntuitionBase *) /* Intuition oeffnen */
63     OpenLibrary("intuition.library",0);
64     if(IntuitionBase == NULL) exit(FALSE);
65
66     GfxBase = (struct GfxBase *) /* Graphics oeffnen */
67     OpenLibrary("graphics.library",0);
68     if(GfxBase == NULL) exit(FALSE);
69
70     if((LScreen=(struct Screen*)
71     OpenScreen(&LowScreen)) == NULL) exit(FALSE);
72
73     if((HScreen=(struct Screen*)
74     OpenScreen(&HighScreen)) == NULL) exit(FALSE);
75
76     SetRGB4(&LScreen->ViewPort,0,15,0,0);
77
78     SetRGB4(&HScreen->ViewPort,0,0,0,15);
79
80     ScreenToFront(LScreen); /* LScreen nach vorne */
81
82     for(zahl=1;zahl<256;++zahl) /* Screens animieren */
83         MoveScreen(LScreen,0,1);
84
85     for(zahl=1;zahl<16;++zahl)
86         MoveScreen(HScreen,0,-4);
87
88     for(zahl=1;zahl<128;++zahl)
89         MoveScreen(LScreen,0,-2);
90
91     WBenchToFront(); /* Workbench nach vorne */
92
93     for(zahl=1;zahl<50000;++zahl);
94
95     CloseWorkBench(); /* Workbench schliessen */
96
97     for(zahl=1;zahl<1000;++zahl)
98         DisplayBeep(NULL); /* Screens Beepen */
99
100     for(zahl=1;zahl<200000;++zahl);
101
```

```

102   OpenWorkBench();                               /* Workbench oeffnen */
103
104   for(zahl=1;zahl<50000;++zahl);
105
106   WBenchToBack();                                 /* Workbench nach hinten */
107
108   for(zahl=1;zahl<500000;++zahl);
109
110   CloseScreen(LScreen);                           /* Screens und Libs schliessen */
111   CloseScreen(HScreen);
112   CloseLibrary(IntuitionBase);
113   CloseLibrary(GfxBase);
114 }

1  /*****
2
3      Halbbrite-Demonstration
4      last update 26/05/87
5      von Frank Krenser und Joerg Koch
6      (c) Markt & Technik 1987
7      *****/
8
9  Diese Demonstration zeigt, wie der Extrahalbbrite-Modus angewendet wird.
10 Die Farben 32 bis 63 haben nur die halbe Intensitaet, wie die Farben 0 bis
11 31. Sie koennen nicht etwa als eigene Farbreister gesetzt werden.
12
13 *****/
14
15 #include <exec/types.h>                            /* Einlesen der Include-Files */
16 #include <exec/tasks.h>
17 #include <exec/libraries.h>
18 #include <exec/devices.h>
19 #include <devices/keymap.h>
20 #include <graphics/copper.h>
21 #include <graphics/display.h>
22 #include <graphics/gfxbase.h>
23 #include <graphics/text.h>
24 #include <graphics/view.h>
25 #include <graphics/gels.h>
26 #include <graphics/regions.h>
27 #include <hardware/blit.h>
28 #include <intuition/intuition.h>
29 #include <intuition/intuitionbase.h>
30
31
32 struct GfxBase      *GfxBase;                       /* Lib - Zeiger */
33 struct IntuitionBase *IntuitionBase;
34
35 struct Screen      *colscreen;
36
37 struct NewScreen newscreen =                        /* Screen definieren */
38 {
39     0,
40     0,
41     320,
42     512,
43     6,
44     0,
45     31,
46     LACE:EXTRA_HALFBRITE,
47     CUSTOMSCREEN,
48     NULL,

```

```
49     "Extra-Halfbrite-Demo bei 320 x 512",
50     NULL,
51     NULL
52     };
53
54     struct IntuiMessage *message;
55
56
57     main()
58     (
59     LONG warte;
60     USHORT schleife1;
61     USHORT schleife2;
62
63     if ((IntuitionBase = (struct IntuitionBase *)
64     OpenLibrary("intuition.library", 0)) == 0) exit();
65                                     /* Libs oeffnen */
66     if ((GfxBase = (struct GfxBase *)
67     OpenLibrary("graphics.library", 0)) == 0) exit();
68
69     colscreen = (struct Screen *)OpenScreen(&newscreen);
70     if (colscreen == 0)
71     {
72                                     /* Screen oeffnen */
73     CloseLibrary(GfxBase);
74     CloseLibrary(IntuitionBase);
75     exit();
76     }
77                                     /* Die Farben der ersten 32 Farbreister setzen */
78     for (schleife1 = 0; schleife1 < 8; schleife1++)
79     {
80     SetRGB4(&colscreen->ViewPort,schleife1+24,schleife1*2,
81     schleife1*2,schleife1*2);
82     SetRGB4(&colscreen->ViewPort,schleife1,schleife1*2,0,0);
83     SetRGB4(&colscreen->ViewPort,schleife1+8,0,schleife1*2,0);
84     SetRGB4(&colscreen->ViewPort,schleife1+16,0,0,schleife1*2);
85     }
86                                     /* Die Farben auf den Schirm bringen */
87     for (schleife1 = 0; schleife1 < 8; schleife1++)
88     for (schleife2 = 0; schleife2 < 8; schleife2++)
89     {
90     SetAPen(&colscreen->RastPort,schleife1 + 8 * schleife2);
91     RectFill(&colscreen->RastPort,schleife1*40,schleife2*62+10,
92     schleife1*40+39,schleife2*62+72);
93     }
94     for(warte = 0; warte < 500000; ++warte);
95
96     CloseScreen(colscreen);          /* Screen und Libs schliessen */
97     CloseLibrary(GfxBase);
98     CloseLibrary(IntuitionBase);
99     )
```

```

1  /*****
2
3      HAM-Demonstration
4      last update 26/05/87
5      von Joerg Koch und Frank Kremser
6      (c) Markt & Technik 1987
7
8  *****/
9
10 Diese Demonstration zeigt alle 4096 Farben des AMIGA auf einmal, wofuer der
11 HAM-Modus verwendet wird.
12
13 *****/
14
15 #include <exec/types.h>
16 #include <exec/tasks.h>
17 #include <exec/libraries.h>
18 #include <exec/devices.h>
19 #include <devices/keymap.h>
20 #include <graphics/copper.h>
21 #include <graphics/display.h>
22 #include <graphics/gfxbase.h>
23 #include <graphics/text.h>
24 #include <graphics/view.h>
25 #include <graphics/gels.h>
26 #include <graphics/regions.h>
27 #include <hardware/blit.h>
28 #include <intuition/intuition.h>
29 #include <intuition/intuitionbase.h>
30
31 struct GfxBase *GfxBase; /* Lib-Zeiger */
32 struct IntuitionBase *IntuitionBase;
33
34 struct RastPort *rp;
35 struct ViewPort *vp;
36
37 struct Screen *screen;
38
39 struct NewScreen newscreen = /* New Screen Demo */
40 {
41     0, /* Linke Ecke */
42     0, /* Obere Ecke */
43     320, /* Breite */
44     256, /* Hoehe */
45     6, /* Tiefe */
46     0, /* DetailPen */
47     1, /* BlockPen */
48     HAM, /* Hold and Modify ViewMode */
49     CUSTOMSCREEN, /* Screen - Typ */
50     NULL,
51     NULL,
52     NULL
53 };
54
55
56 main()
57 {
58     LONG warte;
59     SHORT x, y, r, g, b; /* oeffnen der Libs */
60
61     GfxBase = (struct GfxBase *)OpenLibrary("graphics.library", 0);
62     if (GfxBase == NULL) exit();
63
64     IntuitionBase = (struct IntuitionBase *)OpenLibrary("intuition.library", 0);

```

```
65     if (IntuitionBase == NULL)
66     {
67         CloseLibrary(GfxBase);
68         exit();
69     }                               /* HAM - Screen oeffnen */
70
71     screen = (struct Screen *)OpenScreen(&newscreen);
72     if (screen == NULL)
73     {
74         CloseLibrary(IntuitionBase);    /* Wenn Fehler -> ENDE */
75         CloseLibrary(GfxBase);
76         exit();
77     }
78
79     vp = &screen->ViewPort;
80     rp = &screen->RastPort;
81
82     SetRGB4(vp, 0, 0, 0, 0);
83
84     for(r = 0; r < 16; r++)            /* R-, G- und B-Farbenspiel beginnen */
85     for(g = 0; g < 16; g++)            /* HAM farben, durch halten der 5. */
86     for(b = 0; b < 16; b++)            /* und 6. BitPlane, darstellen */
87     {                                    /* Genaue Beschreibung der HAM- */
88         x = r * 20;                    /* finden Sie bei Kapitel 2.1 */
89         y = g + (b * 16);              /* Die ViewModes !!!!! */
90         SetDrMd(rp, JAM1);
91
92         SetAPen(rp, r + 0x20);
93         Move(rp, x, y);
94         Draw(rp, x, y);
95
96         x++;
97         SetAPen(rp, g + 0x30);
98         Move(rp, x, y);
99         Draw(rp, x, y);
100
101         x++;
102         SetAPen(rp, b + 0x10);
103         Move(rp, x, y);
104         Draw(rp, x + 17, y);
105     }                                    /* Schleife */
106
107     for(warte = 0; warte < 1000000; warte++);
108
109     CloseScreen(screen);                /* Libs und Screen schliessen */
110     CloseLibrary(IntuitionBase);
111     CloseLibrary(GfxBase);
112 }
```

Das Window

Das Window ist neben dem Screen das wohl wichtigste Element für die Programmierung, da im Normalfall sämtliche Ein-/Ausgabeoperationen über Windows ablaufen. Intuition stellt dem Programmierer mit den Windows eine Vielzahl von Möglichkeiten zur Verfügung, ohne daß dieser sich um das Handling zu kümmern braucht. So kann man für sein Programm einfach ein Window öffnen, und ist damit vollkommen unabhängig von weiteren Programmen, die eventuell gleichzeitig ablaufen. Auch kann der Benutzer des Programms später selbst das Window positionieren oder aber in den Hintergrund »klicken«, ohne daß der Programmierer zuvor sein Programm auf all diese Fälle vorbereitet haben muß, da Intuition mit dem Öffnen des Windows die Kontrolle der Systemgadgets übernimmt.

Aber Windows erleichtern nicht nur das Handling, sondern ermöglichen auch eine Vielzahl von Besonderheiten. So kann der Benutzer das Window beliebig vergrößern, verkleinern oder beiseite »legen«. Aber auch für den Programmierer bieten sich einige Vorzüge. So kann der Programmierer über Windows eigene Gadgets verwenden, was die Bedienung eines Programms erheblich vereinfacht. Auch die Pull-Down-Menüs, die der Programmierer über Windows in seinem Programm verwenden kann, vereinfachen die Kommunikation des Benutzers mit dem Programm sehr.

Zudem ist ein Programm, das mit Gadgets und Menüs arbeitet, um ein Vielfaches attraktiver, als ein vergleichbares Programm, das die Eingaben über die Tastatur verlangt.

Es können so viele Windows geöffnet werden, wie der Speicherplatz des Amiga es erlaubt. Im Normalfall dürfte aber ein Window pro Programm genügen. Mehrere Windows sind nötig, wenn mehrere Programme gleichzeitig ablaufen (Multitasking) oder wenn das Programm Ein- und Ausgabe trennen will. Wie viele Windows geöffnet werden müssen, hängt also ganz vom Aufgabengebiet ab.

3.1 Window-Typ

Dem Programmierer stellt Intuition verschiedene Möglichkeiten für Windows zur Verfügung:

Normal-Window:

Dies ist kein spezielles Window mit besonderen Fähigkeiten. Dieses Window wird im Normalfall immer benutzt. Ein Beispiel für dieses Window ist die Workbench-Uhr.

Borderless-Window:

Dieses Window hat im Prinzip die gleiche Funktion wie das normale Window. Der einzige Unterschied ist, daß bei diesem Window keine Randbegrenzungen gezeichnet werden, die in bestimmten Situationen störend sein können.

Gimmezerozero-Window:

Auch dieses Window hat im Prinzip die gleiche Funktion wie ein normales Window. Der Unterschied besteht jedoch darin, daß ein solches Window aus zwei Teilen besteht. Das normale Window wird mitsamt dem Inhalt auf den Screen gezeichnet, benötigt also keinen zusätzlichen Speicherplatz, während der Inhalt beim Gimmezerozero-Window getrennt vom Window-Rand gespeichert wird. Das bedeutet, daß bei einem Gimmezerozero-Window nicht darauf geachtet werden muß, ob über den Rand gezeichnet wird, was bei einem normalen Window schon mal passieren kann.

Zur Kontrolle eines Gimmezerozero-Windows stehen dem Programmierer in der Window-Structure einige spezielle Variablen zur Verfügung:

GZZMouseX und GZZMouseY enthalten die Mauskoordinaten relativ zum inneren Windowbereich.

GZZWidth und GZZHeight enthalten die momentane Breite und Höhe des inneren Windowbereiches in Pixel.

SuperBitMap-Window:

Das SuperBitMap-Window ist das aufwendigste von allen Window-Typen. Für dieses Window muß ein eigenes BitMap erstellt werden, da sämtliche Daten in dieses BitMap geschrieben werden. Von dort werden sie auf den Screen kopiert, wodurch das Window völlig unabhängig ist. Das BitMap muß mindestens so groß sein, wie die maximale Größe des Windows, da ständig alle Daten in der BitMap gespeichert werden. Das bedeutet, daß beim Zeichnen in das Window keine Rücksicht auf die momentane Windowgröße genommen werden muß, da das Window eventuell nur einen Ausschnitt aus dem gesamten Bereich zeigt.

Das BitMap wird folgendermaßen erstellt:

```
InitBitMap(BitMapPtr,Tiefe,Breite,Höhe);
```

Dieser Befehl wird an anderer Stelle noch genau erläutert. Hier brauchen Sie nur zu wissen, welche Bedeutung die Parameter haben:

- BitMapPtr ist ein Zeiger vom Typ BitMapPtr, der bei Aufruf der Funktion noch nicht belegt ist. Nach dem Aufruf zeigt er auf die BitMapStructure des BitMap's.
- Tiefe gibt die Anzahl der Bit-Planes an, das heißt die Anzahl der möglichen Farben. Bei einem SuperBitMap-Window auf der Workbench muß das BitMap eine Tiefe von 2 haben, da die Workbench ebenfalls nur eine Bit-Tiefe von 2 besitzt.
- Breite und Höhe geben die Dimensionen der BitMap in Pixels an. Das BitMap muß mindestens so groß sein, wie das Window maximal sein kann.

Anschließend muß noch der Speicherplatz für das BitMap belegt werden. Dies geschieht mit dem Befehl AllocRaster(Breite,Höhe). Die Parameter Breite und Höhe haben die gleiche Bedeutung wie bei InitBitMap.

Das Ganze sieht also folgendermaßen aus:

```
struct BitMap BM;
.
.
main()
{
  INT i;

  InitBitMap(&BM,2,640,200);
  for (i = 0; i < 2; i++)
    if((BM.Planes[i]=AllocRaster(640,200))==0) exit();
.
.
}
```

Die `exit()` Anweisung beendet das Programm, falls nicht genügend Speicherplatz für die BitMaps vorhanden ist.

Anschließend muß in der `NewWindow-Structure` noch der `BitMap-Zeiger &BM` gesetzt werden.

Backdrop-Window:

Dieses Window hat die gleichen Funktionen wie ein normales Window. Die Ausnahme besteht lediglich darin, daß dieses Window nicht vor ein anderes gelegt werden kann. Das heißt, es erscheint hinter allen anderen Windows, was besonders für "Hintergrund-Informationen" brauchbar ist.

3.2 Window-Refreshing

Intuition stellt dem Programmierer vier Möglichkeiten für das Erneuern seines Windows zur Verfügung:

- NOCAREREFRESH bedeutet, daß das Window nicht erneuert zu werden braucht. Dies ist die einfachste Möglichkeit.
- SIMPLEREFRESH bedeutet, daß Intuition dem Programm über IDCMP mitteilt, daß das Window erneuert werden muß. Anschließend muß Intuition mit BeginRefresh in den Refresh-Zustand versetzt werden. Nun muß das Window vom Programm aus erneuert werden. Zum Schluß muß Intuition mit EndRefresh wieder zurückgesetzt werden. Der Vorteil dieser Methode ist, daß kein zusätzlicher Speicherplatz verwendet wird. Ein Beispiel für diese Methode ist der Editor »ed«.
- SMARTREFRESH reserviert für den momentan sichtbaren Bereich des Windows Speicherbereich und kopiert dort diesen Teil hinein. Wenn das Window zwischenzeitlich von einem anderen Window überlappt wird, kann es im nachhinein wieder restauriert werden, wenn das andere Window »beiseite geschoben« wird. Allerdings kann immer nur der betreffende Ausschnitt erneuert werden. Das heißt, wird das Window vergrößert, können die zuvor nicht sichtbaren Teile nicht restauriert werden. Der Vorteil dieser Methode ist das optimale Speicherplatz/Aufwand-Verhältnis.

Wenn das Window vergrößert wurde, übermittelt Intuition über IDCMP dem Programm die Nachricht, die verbleibenden Bereiche selbst zu erneuern.

- SUPERBITMAP wurde schon oben erläutert. Der Vorteil dieser Methode ist, daß das gesamte Window erneuert wird, also auch die Teile, die vor dem Vergrößern des Windows nicht sichtbar waren. Der Nachteil liegt aber auf der Hand. Es ist der immense Speicherplatzbedarf.

3.3 Die Window-Gadgets

Der Programmierer kann für sein Window einige Funktionen direkt vom Intuition steuern lassen. Die Steuerung diese Funktionen erfolgen über Systemgadgets.

Folgende Systemgadgets stellt Intuition zur Verfügung:

1. **WINDOWDRAG:** Dieses Gadget ist der Balken am oberen Rand des Windows, der es erlaubt, das Window zu verschieben.
2. **WINDOWDEPTH:** stellt die zwei Gadgets dar, mit Hilfe derer das Window in den Bildhintergrund, bzw. Bildvordergrund "gerückt" werden kann.
3. **WINDOWSIZE:** Dieses Gadget ermöglicht die Veränderung der Windowgröße.
4. **WINDOWCLOSE:** Dies ist das einzige Systemgadget, das vom Programm aus abgefragt werden muß, da Intuition das Window nicht selbst schließen kann.

Welche Gadgets verwendet werden sollen, kann der Programmierer frei wählen.

Selbstverständlich kann der Programmierer auch eigene Gadgets "installieren", die er aber auch selbst abfragen muß, wobei Intuition ihn aber hilfreich unterstützt.

3.4 IDCMP

Zuvor wurde schon einige Male von IDCMP gesprochen. Nun wollen wir erläutern, wie IDCMP funktioniert. IDCMP heißt Intuition Direct Communication Message Port.

Ob das Close-Gadget, ein eigenes Gadget oder ein Pull-Down-Menü betätigt wurde, kann das Programm nur von Intuition erfahren. Intuition teilt solche Meldungen über IDCMP dem Programm mit. Das Programm muß diese Meldungen annehmen und beantworten. Erst dann kann die Meldung bearbeitet werden.

```
main()
{
.
.
FOREVER
{
    Wait(1<<WindowPtr->UserPort->mp_SigBit); /*Warten auf Meldung
                                                von Intuition*/
    while (message = GetMsg(WindowPtr->UserPort))
    {
        class = message->Class;           /*Meldung halten*/
        ReplyMsg(message);                /*Meldung bestätigen*/

        if(class==CLOSEWINDOW) exit();    /*Meldung interpretieren*/
    }
.
.
}
```

3.5 Die Window-Befehle

Intuition stellt dem Programmierer eine Vielzahl leistungsfähiger Befehle zur Verfügung. So kann der Programmierer sämtliche Einstellungen des Windows, die er beim Öffnen des Windows angegeben hat jederzeit modifizieren.

Er kann einem Window nachträglich das »WINDOWCLOSE«-Gadget »anhängen« oder aber den Window-Titel ändern.

Zudem kann der Programmierer zu jeder Zeit die Funktion eines System-Gadgets nachahmen, wie zum Beispiel die Größenveränderung mittels »SizeWindow«.

Um ein Window öffnen zu können, muß zuvor eine NewWindow-Structure initialisiert werden. Diese sieht folgendermaßen aus:

```
struct NewWindow
{
    SHORT LeftEdge, TopEdge;      Linke, obere Ecke
    SHORT Width, Height;         Breite und Höhe
    UBYTE DetailPen, BlockPen;   Farben für Gadgets und Rand
    ULONG IDCMPFlags;           Intuition-Meldungen
    ULONG Flags;                 SystemGadgets/Refreshing usw.
    struct Gadget *FirstGadget;  Zeiger auf erstes eigenes Gadget
    struct Image *CheckMark;     Zeiger auf Image für Menüs
    UBYTE *Title;                Window-Titel
    struct Screen *Screen;       Zeiger auf Screen
    struct BitMap *BitMap;       BitMap-Zeiger für SUPERBITMAP
    SHORT MinWidth, MinHeight;   Kleinste Breite/Höhe für Window
    SHORT MaxWidth, MaxHeight;   Größte Breite/Höhe für Window
    USHORT Type;                 Screen-Typ
};
```

Beispiele für die Anwendung finden Sie in einigen Demonstrationen dieses Buches.

Die Parameter im Einzelnen:

| | |
|-----------|---|
| LeftEdge | ein ganzzahliger Wert zwischen 0 und 639, bzw. 319. |
| TopEdge | ein ganzzahliger Wert zwischen 0 und 511, bzw. 255. |
| Width | ein ganzzahliger Wert zwischen 0 und 639, bzw. 319. |
| Height | ein ganzzahliger Wert zwischen 0 und 511, bzw. 255. |
| DetailPen | Farbregister für Detail-Zeichnungen. |
| BlockPen | Farbregister für Flächenzeichnungen. |

IDCMPFlags Hier wird festgesetzt, welche Meldungen von Intuition an das Programm übergeben werden sollen. Folgende Meldungen können übergeben werden. (In der structure müssen sie groß geschrieben werden!):

- Requerify,
- Reqclear,
- Reqset,
- Menuverify,
- Sizeverify,
- Newsized - Meldung, nach einer Größenveränderung des Windows
- Refreshwindow - Meldung, ob Window erneuert werden muß
- Mousebuttons - Meldung, ob Maustasten betätigt worden sind
- Mousemove - Funktioniert nur mit dem Flag REPORTMOUSE und übergibt Meldungen, wenn die Maus bewegt wurde.
- Gadgetdown - Meldung, ob Gadget gedrückt worden ist
- Gadgetup - Meldung, ob Gadget gedrückt und wieder losgelassen wurde
- Menupick - Übergibt die Meldung, welches Menü angewählt wurde
- Closewindow - Meldung, daß das Close-Gadget betätigt wurde
- Rawkey, Vanillakey,
- Intuiticks,
- Activewindow - Meldung, ob Window aktiv ist
- Inactivewindow - Meldung, ob Window inaktiv ist

Nur die wichtigen Flags sind erklärt worden!

Flags: Hier können die verschiedenen Window-Modi eingestellt werden. Folgende Modi stehen zur Verfügung:

- Windowsizing - Systemgadget zum Vergrößern /Verkleinern von Windows

| | | |
|----------------|---|---|
| Sizebright | - | Das Gadget wird an der rechten Seite »angebracht« (Normalstellung) |
| Sizebbottom | - | Das Gadget wird an der unteren Seite »angebracht« (Volle 80 Zeichen) |
| Windowdepth | - | Das Window kann in den Hintergrund »geklickt« werden |
| Windowclose | - | Gadget zum Schließen des Windows |
| Windowdrag | - | Zum Verschieben des Windows |
| Nocarerefresh | - | Kein Erneuern des Windows |
| Simplerefresh | - | Refresh-Typ |
| Smartrefresh | - | Refresh-Typ |
| Superbitmap | - | RefreshTyp |
| Gimmezero-zero | - | Window-Typ |
| Backdrop | - | Window-Typ |
| Borderless | - | Window-Typ |
| Reportmouse | - | Informiert über die Mausposition relativ zur linken oberen Ecke des Windows |
| Activate | - | Das Window wird nach dem Öffnen aktiviert |
| Rmbtrap | - | Setzt die normale Menüsteuerung außer Kraft, sondern steuert sie über die Maustasten |
| FirstGadget: | | Zeiger auf die Gadget-Structure des ersten User-Gadgets. Falls kein eigenes Gadget verwendet wird, wird dieser Parameter gleich NULL gesetzt |
| CheckMark: | | Zeiger auf ein Image für ein User-Häkchen bei Menüs. NULL, wenn das Standard-Häkchen verwendet werden soll. |
| Title: | | Windowtitel. Falls kein Titel gewünscht wird, muß hier NULL eingesetzt werden |
| Screen: | | Wenn ein eigener Screen verwendet wird, also Type gleich CUSTOMSCREEN gesetzt wird, muß hier der Zeiger auf die Screen-Structure stehen, ansonsten NULL |

| | |
|------------|--|
| BitMap: | Zeiger auf ein BitMap-Structure, wenn ein SUPERBITMAP-Window verwendet wird, ansonsten NULL |
| MinWidth: | Minimale Window-Breite |
| MinHeight: | Minimale Window-Höhe |
| MaxWidth: | Maximale Window-Breite |
| MaxHeight: | Maximale Window-Höhe |
| Type: | Screen-Typ: WBENCHSCREEN oder CUSTOMSCREEN ScreenType1 bis 3 sind Intuition-spezifische Screens |

Diese Structure sieht sehr aufwendig aus, doch wenn man sich die Demonstrationen genau ansieht, sollte doch das meiste klar werden. Die Feinheiten werden sowieso erst bei größerer Programmiererfahrung erkennbar.

Nach dem Öffnen eines Windows wird die Window-Structure zurückgegeben, die für die Anwendung der Window-Befehle überaus wichtig ist. Diese Structure führen wir hier nicht ausführlich auf, da ihr Aufbau nicht von größerer Bedeutung ist. Sie beinhaltet folgende zusätzliche Informationen:

- WScreen: Zeiger auf die Structure des zugehörigen Screens
- RPort: Zeiger auf den RastPort des Windows. Dieser Zeiger ist besonders für Textbefehle wichtig

Zugreifen kann man auf diese Daten folgendermaßen:

```
WindowPtr->WScreen und WindowPtr->RPort
```

Nun aber zu den Window-Befehlen:

3.5.1 ActivateWindow

| | |
|------------|---|
| Syntax: | ActivateWindow(WindowPtr); |
| Funktion: | Aktiviert ein Window |
| Parameter: | WindowPtr -> Zeiger auf die Window-Structure des Windows, das aktiviert werden soll |
| Ergebnis: | Kein Ergebnis. |
| Datentyp: | struct Window *WindowPtr; |
| Sonstiges: | Das Window kann zu Beginn, also gleich nach dem Öffnen aktiviert werden, indem das ACTIVATE-Flag in der NewWindow-Structure gesetzt wird. |

Referenz: Siehe auch die Flags der NewWindow-Structure.

3.5.2 BeginRefresh

Syntax: `BeginRefresh(WindowPtr);`

Funktion: Diese Routine setzt Intuition und die Layer-Library in einen »Refresh-Zustand«. Wenn der Programmierer sein Window nun Stück für Stück erneuert, ignoriert Intuition sämtliche Befehle, die sich auf Bereiche außerhalb des Windows beziehen. Dadurch wird eine enorme Effizienz erzielt.

Parameter: `WindowPtr` -> Dieser Pointer zeigt auf das Window-Structure des Windows, das erneuert werden soll.

Ergebnis: Kein Ergebnis.

Datentyp: `struct Window *WindowPtr;`

Sonstiges: Besonders effizient läßt sich ein Window erneuern, wenn man es zuvor als »SIMPLE_REFRESH«-Window geöffnet hat. Ist das Window teilweise zerstört, beispielsweise durch zeitweise Überlagerung eines anderen Windows, so erhält das Programm von Intuition die Meldung, das Window zu erneuern (»refresh«). Nun gibt das Programm den Befehl »BeginRefresh(..)« und beginnt den Inhalt des Windows selbstständig zu erneuern. Anschließend muß noch der Befehl »EndRefresh« gegeben werden, um Intuition wieder in den Normal-Modus zurückzusetzen.

Diese Refresh-Methode ist besonders bei Windows empfehlenswert, die Texte enthalten, die das Programm gespeichert hat. Würde man für solch eine Anwendung ein `SUPER_BITMAP_WINDOW` öffnen, wäre dies eine unnötige Speicherplatzverschwendung.

Referenz: Siehe auch `EndRefresh`.

3.5.3 ClearPointer

Syntax: `ClearPointer(WindowPtr);`

Funktion: Löscht den selbst definierten Mauszeiger.

Parameter: `WindowPtr` -> Dieser Pointer zeigt auf das Window-Structure des Windows, dessen Mauszeiger gelöscht werden soll.

- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct Window *WindowPtr; Sonstiges: Mit dem Befehl »SetPointer« kann jedem Window ein eigener Mauszeiger zugewiesen werden, der immer dann sichtbar wird, wenn das betreffende Window aktiv, das heißt »angeklickt« ist. Dieser Zeiger kann mit dem Befehl »ClearPointer« wieder gelöscht werden. Dann ist wieder der normale Intuition-Zeiger -Pfeil-sichtbar.
- Referenz:** Siehe auch SetPointer.

3.5.4 CloseWindow

- Syntax:** CloseWindow(WindowPtr);
- Funktion:** Schließt ein Window, das zuvor mit OpenWindow geöffnet worden ist.
- Parameter:** WindowPtr -> Dieser Pointer zeigt auf das Window-Structure des Windows, das geschlossen werden soll.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct Window *WindowPtr;
- Sonstiges:** Dieser Befehl schließt das betreffende Window und gibt den Speicherplatz frei.
- Der Programmierer muß dabei jedoch beachten, daß er zuvor alle Menüs, die er für dieses Window erstellt hat, mit »ClearMenuStrip« löscht, da Intuition ansonsten eventuell »abstürzt«!
- Zudem müssen alle Anfragen von Intuition über IDCMP mit »Reply« beantwortet worden sein, falls man einen IDCMP geöffnet hat. Ist dies nicht der Fall, übergeht Intuition alle Meldungen, die in Folge auftreten.
- Referenz:** Siehe auch OpenWindow.

3.5.5 EndRefresh

- Syntax:** EndRefresh(WindowPtr,Voll);
- Funktion:** Diese Routine setzt Intuition und die Layer-Library vom »Refresh-Zustand« zurück in den Normal-Modus. Der »Refresh-Zustand« muß zuvor mit »BeginRefresh« eingestellt worden sein.
- Parameter:** WindowPtr -> Dieser Pointer zeigt auf das Window-Structure des Windows, das erneuert worden ist.
- Voll -> enthält nach dem Aufruf einen Wahrheitswert, TRUE oder FALSE. Ist der Wert TRUE, so wurde das Window vollständig erneuert, ist er FALSE, ist das Window nicht komplett erneuert worden.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct Window *WindowPtr;
bool Voll;
- Sonstiges:** Besonders effizient läßt sich ein Window erneuern, wenn man es zuvor als »SIMPLE_REFRESH«-Window geöffnet hat. Ist das Window teilweise zerstört, beispielsweise durch zeitweise Überlagerung eines anderen Windows, so erhält das Programm von Intuition die Meldung, das Window zu erneuern (»refreshen«). Nun gibt das Programm den Befehl »BeginRefresh(..)« und beginnt den Inhalt des Windows selbständig zu erneuern. Anschließend muß der Befehl »EndRefresh« gegeben werden, um Intuition wieder in den Normal-Modus zurückzusetzen.
- Referenz:** Siehe auch BeginRefresh.

3.5.6 ModifyIDCMP

- Syntax:** `ModifyIDCMP(WindowPtr, IDCMPFlags);`
- Funktion:** Dieser Befehl ändert den Zustand des Window-IDCMP.
- Parameter:** `WindowPtr` -> Dieser Pointer zeigt auf das Window-Structure des Windows, dessen Informationsport zum Intuition aktualisiert werden soll.
- `IDCMPFlags` -> gibt den neuen Zustand des IDCMP an.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** `struct Window *WindowPtr;`
`WORD IDCMPFlags;`
- Sonstiges:** Über IDCMP – Intuition Direct Communication Message Port – kommunizieren Intuition und Programm. So teilt Intuition dem Programm über IDCMP beispielsweise mit, daß das WINDOWCLOSE-Gadget betätigt wurde. Dies allerdings nur, wenn zuvor das IDCMP-Flag CLOSEWINDOW gesetzt wurde. Soll nun ein eigenes, nachträglich mit »AddGadget« eingefügtes, Gadget abgefragt werden, so muß dies ebenfalls über IDCMP geschehen. Dazu muß Intuition allerdings mitgeteilt werden, daß es auch diese Information übermitteln soll. Hier wird nun der Befehl »ModifyIDCMP« verwendet. In unserem Beispiel folgendermaßen:
- ```
ModifyIDCMP(WindowPtr, GADGETUP | GADGETDOWN);
```
- Dem Programm wird nun mitgeteilt, ob ein Gadget betätigt wurde.
- Referenz:** Für die IDCMPFlags siehe auch unter Kap. 3 das Window

### 3.5.7 MoveWindow

|            |                                                                                                                                         |                                                                                                                                                               |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | <code>MoveWindow(WindowPtr,dx,dy);</code>                                                                                               |                                                                                                                                                               |
| Funktion:  | Verschiebt ein Window um eine bestimmte Anzahl von Punkten.                                                                             |                                                                                                                                                               |
| Parameter: | <code>WindowPtr</code>                                                                                                                  | -> Dieser Pointer zeigt auf das Window-Structure des Windows, das verschoben werden soll.                                                                     |
|            | <code>dx</code>                                                                                                                         | -> gibt an, um wieviele Punkte das Window nach rechts oder links verschoben werden soll. Ist <code>dx &gt; 0</code> , wird daß Window nach rechts verschoben. |
|            | <code>dy</code>                                                                                                                         | -> gibt an, um wieviele Punkte das Window nach oben oder unten verschoben werden soll. Ist <code>dy &gt; 0</code> , wird daß Window nach unten verschoben.    |
| Ergebnis:  | Kein Ergebnis.                                                                                                                          |                                                                                                                                                               |
| Datentyp:  | <code>struct Window *WindowPtr;</code><br><code>BYTE dx, dy;</code>                                                                     |                                                                                                                                                               |
| Sonstiges: | Sollten <code>dx</code> oder <code>dy</code> zu groß oder zu klein gewählt worden sein, wird das Window so weit wie möglich verschoben. |                                                                                                                                                               |
| Referenz:  | Siehe auch <code>SizeWindow</code><br><code>WindowToFront</code><br><code>WindowToBack</code>                                           |                                                                                                                                                               |

### 3.5.8 OpenWindow

|            |                                                                                                                                                                  |                                                                                                                             |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | <code>Window = OpenWindow(NewWindow);</code>                                                                                                                     |                                                                                                                             |
| Funktion:  | Öffnet ein Window mit den Werten, die in der <code>NewWindow</code> -Structure enthalten sind und gibt den Zeiger auf das <code>Window</code> -Structure zurück. |                                                                                                                             |
| Parameter: | <code>NewWindow</code>                                                                                                                                           | -> ist die <code>NewWindow</code> -Structure, die übergeben werden muß. Zuvor muß sie allerdings initialisiert worden sein. |
| Ergebnis:  | <code>Window</code>                                                                                                                                              | -> Zeiger auf die <code>Window</code> -Structure des geöffneten Windows.                                                    |
| Datentyp:  | <code>struct NewWindow NewWindow;</code><br><code>struct Window *Window;</code>                                                                                  |                                                                                                                             |

**Sonstiges:** Nachdem das Window geöffnet worden ist, kann die NewWindow-Structure für weitere Windows verwendet werden, da sie nicht mehr von Bedeutung ist.

Der Zeiger auf die neue Window-Structure muß allerdings übernommen werden, da er noch benötigt wird. Dazu ein Beispiel:

```

..
struct Window ownwindow;
..
struct NewWindow nw =
 { .. };
..
main()
{..
 ownwindow = OpenWindow(nw);
..}

```

**Referenz:** Für den Aufbau der NewWindow- und der Window-Structure siehe unter Kap. 3 "Das Window".

### 3.5.9 RefreshWindowFrame

**Syntax:** RefreshWindowFrame(WindowPtr);

**Funktion:** erneuert die Umrandung des spezifizierten Windows

**Parameter:** WindowPtr -> Zeiger auf die Window-Structure des Windows, dessen Umrandung erneuert werden soll.

**Ergebnis:** Kein Ergebnis.

**Datentyp:** struct Window \*WindowPtr;

**Sonstiges:** Im Normalfall muß dieser Befehl nicht gegeben werden, da Intuition dies von selbst handhabt.

**Referenz:** Siehe auch RefreshDisplay

### 3.5.10 ReportMouse

|            |                                                                                                                                                                                                                                                                                                                                                                                |                                                                                                                   |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| Syntax:    | ReportMouse(WindowPtr,Status);                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                   |
| Funktion:  | Dieser Befehl setzt das »REPORTMOUSE«-Flag in der Window-Structure, bzw. setzt es zurück. Ist es gesetzt, wird das Programm bei aktiviertem Window ständig von Intuition über die Position der Maus informiert.                                                                                                                                                                |                                                                                                                   |
| Parameter: | WindowPtr                                                                                                                                                                                                                                                                                                                                                                      | -> Dieser Pointer zeigt auf die Window-Structure des Windows, der die Maus-Informationen übergeben werden sollen. |
|            | Status                                                                                                                                                                                                                                                                                                                                                                         | -> ist ein Wahrheitswert. Wird TRUE angegeben, wird die Funktion eingeschaltet. FALSE schaltet sie aus.           |
| Ergebnis:  | Kein Ergebnis.                                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                   |
| Datentyp:  | struct Window *WindowPtr;<br>bool Status;                                                                                                                                                                                                                                                                                                                                      |                                                                                                                   |
| Sonstiges: | Wird dieser Befehl verwendet, wenn zur gleichen Zeit ein Gadget vom Benutzer aktiviert, das heißt » angeklickt« ist, bleibt er nur so lange aktiv, bis das Gadget deaktiviert wird. Dies ist besonders bei Gadgets wichtig, die mit der Maus verschoben werden können, wie zum Beispiel das SIZE-Gadget. Damit kann das Programm ständig die Gadget-Bewegungen nachvollziehen. |                                                                                                                   |
| Referenz:  | Siehe auch die Flags der NewWindow-Structure                                                                                                                                                                                                                                                                                                                                   |                                                                                                                   |

### 3.5.11 SetPointer

|            |                                                 |                                                                                                              |
|------------|-------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| Syntax:    | SetPointer(WindowPtr,&Data[0],Höhe,Breite,X,Y); |                                                                                                              |
| Funktion:  | Setzt einen Window-spezifischen Mauszeiger.     |                                                                                                              |
| Parameter: | WindowPtr                                       | -> Dieser Pointer zeigt auf die Window-Structure des Windows, das einen speziellen Mauszeiger erhalten soll. |
|            | &Data[0]                                        | -> ist ein Zeiger auf die Sprite-Daten für den neuen Zeiger.                                                 |
|            | Höhe                                            | -> gibt die Höhe des Zeigers in Punkten an.                                                                  |

- Breite -> gibt die Breite des Zeigers in Punkten an. Maximal darf das Sprite 16 Punkte breit sein.
- X,Y -> geben die relative Koordinate des Aktiv-Punktes an. Die Koordinate muß relativ zur linken, oberen Ecke des Sprites angegeben werden, wobei darauf zu achten ist, daß die Koordinaten negativ einzugeben sind!

Ergebnis: Kein Ergebnis.

Datentyp: struct Window \*WindowPtr;  
USHORT Data[];  
int Höhe, Breite, X, Y;

Sonstiges: Der normale Mauszeiger –der Pfeil– wechselt immer dann in den selbst definierten, wenn das spezialisierte Window aktiv ist. Sobald es deaktiviert ist, wird wieder der normale Mauszeiger sichtbar.

Der Aktiv-Punkt gibt die Stelle des Zeigers an, mit der der Benutzer über einem Gadget o.ä. sein muß, wenn er es »betätigen« will.

Für X und Y können auch positive Zahlen eingegeben werden, was aber nicht sehr sinnvoll ist, da der Aktiv-Punkt dann außerhalb des Sprites, bzw. des Zeigers liegt. Dies ist auch der Fall, falls X kleiner als -15 ist, oder Y kleiner als das Negativ von der Zeigerhöhe.

Beispiel für die Deklaration:

```
USHORT data[] =
{
 0x0000 0x0000 /* Start-Bytes müssen immer 0 sein*/

 0xFFFF 0xFFFF 0xF00F 0xF00F 0xF00F 0xF00F 0xF00F 0xF00F
 0xF00F 0xF00F 0xF00F 0xF00F 0xF00F 0xF00F 0xFFFF 0xFFFF

 0x0000 0x0000 /* End-Bytes müssen immer 0 sein*/
};

main()
{
```

```
SetPointer(WindowPtr,&data[0],16,16,-7,-7);
.
.
}
```

Referenz: Siehe auch ClearPointer und Kapitel 5.3 Animation mit SetPointer

### 3.5.12 SetWindowTitles

Syntax: SetWindowTitles(WindowPtr,&WTitel[0],&STitel[0]);

Funktion: Gibt dem Window einen neuen Namen und setzt einen Window-spezifischen Screen-Titel.

Parameter: WindowPtr -> Dieser Pointer zeigt auf die Window-Structure des Windows, dessen Titel geändert werden sollen.

&WTitel[0] -> Zeiger auf einen neuen Window-Titel.

&STitel[0] -> Zeiger auf einen neuen Screen-Titel.

Ergebnis: Kein Ergebnis.

Datentyp: struct Window \*WindowPtr;  
char WTitel[];  
char STitel[];

Sonstiges: Dieser Befehl setzt einen neuen Window-Titel, der immer in der Window-Titelzeile erscheint. Mit diesem Befehl kann auch der spezielle Screen-Titel geändert werden, der immer dann erscheint, wenn das Window aktiv ist.

Für WTitel und STitel können auch die Werte -1 und 0 eingesetzt werden.

-1 bewirkt keine Änderung des betreffenden Titels, der alte Titel bleibt erhalten. Dies ist dann zu verwenden, wenn nur einer der Titel geändert werden soll.

0 bewirkt die Löschung des betreffenden Titels.

Referenz: Siehe auch unter Kap. 3 Das Window und unter Kap 2. ShowTitle.

### 3.5.13 SizeWindow

|            |                                                                                                                                                                                                                                                                                 |                                                                                                  |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| Syntax:    | <code>SizeWindow(WindowPtr,dx,dy);</code>                                                                                                                                                                                                                                       |                                                                                                  |
| Funktion:  | Modifiziert die Größe eines Windows.                                                                                                                                                                                                                                            |                                                                                                  |
| Parameter: | <code>WindowPtr</code>                                                                                                                                                                                                                                                          | -> Dieser Pointer zeigt auf die Window-Structure des Windows, das modifiziert werden soll.       |
|            | <code>dx</code>                                                                                                                                                                                                                                                                 | -> gibt an, um wieviele Punkte das Window in X-Richtung vergrößert oder verkleinert werden soll. |
|            | <code>dy</code>                                                                                                                                                                                                                                                                 | -> gibt an, um wieviele Punkte das Window in Y-Richtung vergrößert oder verkleinert werden soll. |
| Ergebnis:  | Kein Ergebnis.                                                                                                                                                                                                                                                                  |                                                                                                  |
| Datentyp:  | <code>struct Window *WindowPtr;</code><br><code>int dx, dy;</code>                                                                                                                                                                                                              |                                                                                                  |
| Sonstiges: | Für <code>dx</code> und für <code>dy &lt; 0</code> wird das Window verkleinert.<br><br>Überaus wichtig ist, daß Intuition nicht überprüft, ob die Werte zulässig sind. Das heißt, wenn das Window auf 1000 x 1000 Punkte vergrößert wurde, erhält man sehr unschöne Ergebnisse. |                                                                                                  |
| Referenz:  | Siehe auch <code>MoveWindow</code><br><code>WindowToFront</code><br><code>WindowToBack</code>                                                                                                                                                                                   |                                                                                                  |

### 3.5.14 ViewPortAddress

|            |                                                                                                                        |                                                                                                              |
|------------|------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| Syntax:    | <code>vp = ViewPortAddress(WindowPtr);</code><br>oder<br><code>if (ViewPortAddress(WindowPtr) == 34565)</code><br>usw. |                                                                                                              |
| Funktion:  | Ermittelt die Adresse des ViewPorts des spezifizierten Windows.                                                        |                                                                                                              |
| Parameter: | <code>WindowPtr</code>                                                                                                 | -> Dieser Pointer zeigt auf die Window-Structure des Windows, dessen ViewPort-Adresse ermittelt werden soll. |

|            |                                                                                                                                                                                                                                                             |                                                         |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------|
| Ergebnis:  | vp                                                                                                                                                                                                                                                          | -> Adresse des ViewPorts, auf dem das Window erscheint. |
| Datentyp:  | struct Window *WindowPtr;<br>ULONG vp;                                                                                                                                                                                                                      |                                                         |
| Sonstiges: | Dieser Befehl wird äußerst selten benötigt, da meist der Zeiger auf den RastPort des Windows ausreicht. Sollte eine Funktion benutzt werden, die den ViewPort des Screens, auf dem sich das Window befindet, benötigt, kann dieser Befehl verwendet werden. |                                                         |
| Referenz:  | Für Informationen über View- und Rastport siehe auch Kapitel 2 »Der Screen«.                                                                                                                                                                                |                                                         |

### 3.5.15 WindowLimits

|            |                                                                                                                               |                                                                                                                                                                                                      |
|------------|-------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | ok = WindowLimits(WindowPtr,MinX,MinY,MaxX,MaxY);<br>oder<br>if(WindowLimits(WindowPtr,MinX,MinY,MaxX,MaxY) == FALSE)<br>usw. |                                                                                                                                                                                                      |
| Funktion:  | Setzt die Größenbeschränkungen eines Windows neu.                                                                             |                                                                                                                                                                                                      |
| Parameter: | WindowPtr                                                                                                                     | -> Dieser Pointer zeigt auf die Window-Structure des Windows, dessen Größenbeschränkungen neu gesetzt werden sollen.                                                                                 |
|            | MinX                                                                                                                          | -> setzt die Minimalgröße des Windows in X-Richtung.                                                                                                                                                 |
|            | MinY                                                                                                                          | -> setzt die Minimalgröße des Windows in Y-Richtung.                                                                                                                                                 |
|            | MaxX                                                                                                                          | -> setzt die Maximalgröße des Windows in X-Richtung.                                                                                                                                                 |
|            | MaxY                                                                                                                          | -> setzt die Maximalgröße des Windows in Y-Richtung.                                                                                                                                                 |
| Ergebnis:  | ok                                                                                                                            | -> Gibt den Wahrheitswert TRUE zurück, falls alles in Ordnung war. Waren die Minimalwerte größer oder waren die Maximalwerte kleiner als die derzeitige Größe des Windows, wird FALSE zurückgegeben. |

|            |                                                                                                                                                                                                                                                                                                                                                         |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Datentyp:  | struct Window *WindowPtr;<br>int MinX, MinY, MaxX, MaxY;<br>bool ok;                                                                                                                                                                                                                                                                                    |
| Sonstiges: | Als Wert kann auch 0 angegeben werden, was bewirkt, daß die entsprechende Größenbeschränkung nicht geändert wird, was verwendet wird, um nur einzelne Werte zu ändern.<br><br>Ist ein einzelner Wert nicht mit der derzeitigen Windowgröße vereinbar, wird er ignoriert und es wird FALSE zurückgegeben. Die anderen Werte werden trotzdem modifiziert. |
| Referenz:  | Siehe auch OpenWindow                                                                                                                                                                                                                                                                                                                                   |

### 3.5.16 WindowToBack

|            |                                                                                                                                                                                                                                                                                                           |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | WindowToBack(WindowPtr);                                                                                                                                                                                                                                                                                  |
| Funktion:  | Setzt ein Window hinter alle anderen Windows zurück.                                                                                                                                                                                                                                                      |
| Parameter: | WindowPtr -> Dieser Pointer zeigt auf die Window-Structure des Windows, das zurückgesetzt werden soll.                                                                                                                                                                                                    |
| Ergebnis:  | Kein Ergebnis.                                                                                                                                                                                                                                                                                            |
| Datentyp:  | struct Window *WindowPtr;                                                                                                                                                                                                                                                                                 |
| Sonstiges: | Das spezifizierte Window wird durch diesen Befehl hinter alle anderen Windows zurückgesetzt. Allein die Backdrop-Windows bilden hier eine Ausnahme. Wendet man den Befehl auf ein solches Window an, geschieht nichts, da diese Windows schon zuhinterst sind und zudem nicht hervorgeholt werden können. |
| Referenz:  | Für Backdrop-Windows siehe auch Kap. 3 »Das Window«. Siehe auch MoveWindow<br>SizeWindow<br>WindowToFront                                                                                                                                                                                                 |

### 3.5.17 WindowToFront

|            |                                                                                                       |
|------------|-------------------------------------------------------------------------------------------------------|
| Syntax:    | WindowToFront(WindowPtr);                                                                             |
| Funktion:  | Setzt ein Window vor alle anderen Windows.                                                            |
| Parameter: | WindowPtr -> Dieser Pointer zeigt auf die Window-Structure des Windows, das hervorgeholt werden soll. |

- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct Window \*WindowPtr;
- Sonstiges:** Das spezifizierte Window wird durch diesen Befehl vor alle anderen Windows gesetzt. Allein die Backdrop-Windows bilden hier eine Ausnahme. Wendet man den Befehl auf ein solches Window an, geschieht nichts, da diese Windows nicht hervorgeholt werden können.
- Referenz:** Für Backdrop-Windows siehe auch Kap. 3 Das Window.  
Siehe auch MoveWindow  
SizeWindow  
WindowToBack

```
1 /*****
2
3 Window-Demonstration
4 last update 26/05/87
5 von Frank Kremser und Joerg Koch
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Diese Demonstration zeigt die Anwendung der Windowbefehle
11
12 *****/
13
14 #include <exec/types.h> /* Include-Files einladen */
15 #include <exec/nodes.h>
16 #include <exec/lists.h>
17 #include <exec/ports.h>
18 #include <exec/devices.h>
19 #include <devices/keymap.h>
20 #include <graphics/regions.h>
21 #include <graphics/copper.h>
22 #include <graphics/gels.h>
23 #include <graphics/gfxbase.h>
24 #include <graphics/gfx.h>
25 #include <graphics/clip.h>
26 #include <graphics/view.h>
27 #include <graphics/rastport.h>
28 #include <graphics/layers.h>
29 #include <intuition/intuition.h>
30 #include <hardware/blit.h>
31
32 struct IntuitionBase *IntuitionBase; /* Lib - Zeiger */
33 struct GfxBase *GfxBase;
34
35 struct RastPort *rp;
36 struct Window *nor, *bor, *bac;
37 struct IntuiMessage *message;
38
39 struct NewWindow normal =
40 {
41 0, /* Linke Ecke */
42 10, /* Obere Ecke */
43 200, /* Breite */
44 100, /* Hoehe */
```

```
45 3, /* DetailPen */
46 1, /* BlockPen */
47 CLOSEWINDOW:REFRESHWINDOW, /* IDCMP-Flags */
48 WINDOWCLOSE:WINDOWSIZING:WINDOWDEPTH:WINDOWDRAG:NOCAREREFRESH, /* Flags */
49 NULL,
50 NULL,
51 "Normal-Window", /* Window-Text */
52 NULL,
53 NULL,
54 20,
55 20,
56 640,
57 256,
58 WBENCHSCREEN /* Screen-Typ */
59);
60
61 struct NewWindow backdrop =
62 {
63 220,
64 80,
65 200,
66 100,
67 3,
68 1,
69 CLOSEWINDOW:REFRESHWINDOW,
70 NOCAREREFRESH:BACKDROP,
71 NULL,
72 NULL,
73 "Backdrop-Window",
74 NULL,
75 NULL,
76 0,
77 0,
78 0,
79 0,
80 WBENCHSCREEN
81 };
82
83 struct NewWindow borderless =
84 {
85 440,
86 10,
87 200,
88 100,
89 3,
90 1,
91 CLOSEWINDOW:REFRESHWINDOW,
92 WINDOWSIZING:WINDOWDEPTH:WINDOWDRAG:NOCAREREFRESH:BORDERLESS,
93 NULL,
94 NULL,
95 "Borderless-Window",
96 NULL,
97 NULL,
98 20,
99 20,
100 640,
101 256,
102 WBENCHSCREEN
103 };
104
105 USHORT NormImage[] = /* SpriteImage-Structure */
106 {
107 0,0,
108
```

```
109 0xFFFE, 0xB001,
110 0xB000, 0xB001,
111 0xB000, 0xB001,
112 0xB000, 0xB001,
113 0xB000, 0xB001,
114 0xB000, 0xB001,
115 0xB000, 0xB001,
116
117 0,0
118);
119
120 USHORT BorImage[] = /* SpriteImage-Structure */
121 {
122 0,0,
123
124 0xC003, 0x0000,
125 0xE007, 0x0000,
126 0x700E, 0x0000,
127 0x3B1C, 0x0000,
128 0x1C3B, 0x0000,
129 0x0E70, 0x0000,
130 0x07E0, 0x0000,
131 0x03C0, 0x0000,
132 0x03C0, 0x0000,
133 0x07E0, 0x0000,
134 0x0E70, 0x0000,
135 0x1C3B, 0x0000,
136 0x3B1C, 0x0000,
137 0x700E, 0x0000,
138 0xE007, 0x0000,
139 0xC003, 0x0000,
140
141 0,0
142);
143
144 USHORT BacImage[] = /* SpriteImage-Structure */
145 {
146 0,0,
147
148 0x0000, 0xC003,
149 0x0000, 0xE007,
150 0x0000, 0x700E,
151 0x0000, 0x3B1C,
152 0x0000, 0x1C3B,
153 0x0000, 0x0E70,
154 0x0000, 0x07E0,
155 0x0000, 0x03C0,
156 0x0000, 0x03C0,
157 0x0000, 0x07E0,
158 0x0000, 0x0E70,
159 0x0000, 0x1C3B,
160 0x0000, 0x3B1C,
161 0x0000, 0x700E,
162 0x0000, 0xE007,
163 0x0000, 0xC003,
164
165 0,0
166);
167
168 main()
169 {
170 ULONG MessageClass;
171 USHORT code;
172 LONG warte;
```

```
173
174 if(!(GfxBase = (struct GfxBase *)OpenLibrary("graphics.library",0))
175 {
176 close_things();
177 exit(); /* oeffnen der Libraries */
178 }
179
180 if(!(IntuitionBase = (struct IntuitionBase *)
181 OpenLibrary("intuition.library",0))
182 {
183 close_things();
184 exit();
185 }
186
187 if (!(bor = (struct Window *)OpenWindow(&borderless)))
188 {
189 close_things(); /* oeffnen der Windows */
190 exit();
191 }
192
193 if (!(bac = (struct Window *)OpenWindow(&backdrop)))
194 {
195 close_things();
196 exit();
197 }
198
199 if (!(nor = (struct Window *)OpenWindow(&normal)))
200 {
201 close_things();
202 exit();
203 }
204
205 SetPointer(nor,&NormImage[0],7,16,-7,-4); /* Setzen des Mauszeigers */
206 SetWindowTitles(nor,-1, /* Window-Titel setzen */
207 "Dies ist der Screen-Titel f[r das normale Window");
208
209 SetPointer(bor,&BorImage[0],16,16,-7,-7);
210 SetWindowTitles(bor,-1,
211 "Dies ist der Screen-Titel f[r das Borderless-Window");
212
213 SetPointer(bac,&BacImage[0],16,16,-7,-7);
214 SetWindowTitles(bac,-1,
215 "Dies ist der Screen-Titel f[r das Backdrop-Window");
216
217 rp = nor->RPort;
218 /* Window verschieben */
219 MoveWindow(nor,180,40);
220
221 SizeWindow(nor,50,50); /* Window vergroessern */
222
223 for(warte = 0; warte < 100000; warte++);
224
225 SizeWindow(nor,-50,-50);
226
227 MoveWindow(bor,-180,120);
228
229 for(warte = 0; warte < 10; warte++)
230 {
231 WindowToFront(nor);
232 WindowToFront(bor); /* Window nach vorn */
233 };
234
235 for(;;)
```

```
236 {
237 if (message = (struct IntuiMessage *)GetMsg(nor->UserPort))
238 { /* auf Message von Intuition warten */
239 MessageClass = message->Class; /* Message retten */
240 code = message->Code;
241 ReplyMsg(message);
242 if (MessageClass == CLOSEWINDOW) (close_things(); exit());};
243 }; /* wenn Window geschlossen wurde -> Ende */
244 }
245 }
246
247 close_things() /* Unterroutine zum Schliessen */
248 {
249 CloseWindow(nor); /* Windows und Libs schliessen */
250 CloseWindow(bac);
251 CloseWindow(bor);
252 CloseLibrary(GfxBase);
253 CloseLibrary(IntuitionBase);
254 }
```

# Zeichnen in Screens und Windows

In diesem Kapitel wollen wir die einfachen Zeichenbefehle behandeln, die der Amiga zur Verfügung stellt. Im Gegensatz zu einigen anderen Rechnern ist eine Vielzahl von Grafikbefehlen schon vorhanden, die nur noch aufgerufen werden müssen.

Aber auch Texte können mit einfachen Befehlen auf beliebige Screens oder Windows geschrieben werden. Dabei stehen selbstverständlich auch verschiedene Schriftarten bereit, die wiederum in vielen Varianten dargestellt und kombiniert werden können.

Weitere einfache Grafikmöglichkeiten sind Images und Borders. Images sind beliebig große Grafiken, die an jeder Stelle des Screens plaziert werden können. So verwenden viele Gadgets, die eine bestimmte Form haben sollen, eigene Images.

Borders sind eine Anzahl von Linien, die durch  $x,y$ -Koordinaten festgelegt sind. Wie der Name schon sagt, können damit Ränder gezeichnet werden, oder aber auch ganz andere Dinge, wie zum Beispiel Rechtecke oder Polygone. Für welche Aufgaben der Programmierer sie nutzt, bleibt ihm allein überlassen.

## 4.1 Einfache Zeichenbefehle

Das der Amiga ein Grafiktalent ist, ist gemeinhin bekannt. Aber das diese Grafikmöglichkeiten einfach zu handhaben sind, ist bei weitem nicht so verbreitet.

Die Graphics-Bibliothek des Amiga stellt eine Vielzahl von Befehlen zur Verfügung, die bei vielen anderen Rechnern erst umständlich zu implementieren oder gar nicht realisierbar sind.

Zu der ersten Gruppe zählen die normalen Grafikbefehle, wie DrawCircle, DrawEllipse, Draw, Move usw. Diese Befehle sind von anderen Computern her bekannt.

Zur zweiten Gruppe zählen die Area-Befehle. Die Area-Befehle erleichtern die Handhabung von gefüllten Flächen ungemein. So lassen sich gefüllte Kreise und Ellipsen mittels eines einzigen Befehls erzeugen. Und dies in enormer Geschwindigkeit. Aber auch gefüllte Polygone – Vielecke – lassen sich sehr einfach erstellen. Diese Polygone dürfen beliebig viele Eckpunkte haben.

Allein die Installierung der Areas erscheint umständlich, da ungewohnt:

1. Mit AllocRaster muß Speicherbereich reserviert werden, der mindestens so groß sein muß, wie das größte, zu zeichnende Object. Im Zweifelsfall ist dies die Größe des Screens, auf den gezeichnet wird.
2. Mit InitTmpRas muß ein Speicherbereich für die Verwendung durch Areas installiert werden.
3. Mit InitArea wird angezeigt, daß Area-Befehle verwendet werden.
4. Mit FreeRaster wird am Ende des Programms der belegte Speicherplatz wieder verfügbar gemacht.

Nähere Informationen zu diesem Thema sind dem Demonstrationsprogramm zu entnehmen.

### 4.1.1 AreaCircle

Syntax: fehler = AreaCircle(rastport,x,y,radius);

Funktion: Fügt eine Kreisinformation zur Area-Info-Liste hinzu. Das bedeutet, wenn alle Areas gezeichnet werden, wird auch dieser Kreis gezeichnet und ausgefüllt.

|            |                                                                                      |                                                                                                         |
|------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|
| Parameter: | rastport                                                                             | -> Zeiger auf die RastPort-Structure des Screens oder Windows, in den der Kreis gezeichnet werden soll. |
|            | x,y                                                                                  | -> Koordinaten des Mittelpunktes.                                                                       |
|            | radius                                                                               | -> Radius des Kreises in Pixel.                                                                         |
| Ergebnis:  | fehler                                                                               | -> 0, falls erfolgreich, 1, falls nicht.                                                                |
| Datentyp:  | struct RastPort *rastport;<br>int x, y, radius;<br>int fehler;                       |                                                                                                         |
| Sonstiges: | Der Kreis wird in der Farbe ausgefüllt, die für die momentan definierten Areas gilt. |                                                                                                         |
| Referenz:  | Für Areas siehe InitArea                                                             |                                                                                                         |

### 4.1.2 AreaDraw

|            |                                                                                                                                            |                                                                                                               |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| Syntax:    | fehler = AreaDraw(rastport,x,y);                                                                                                           |                                                                                                               |
| Funktion:  | Fügt einen Eckpunkt für ein Polygon an die Area-Liste ein.                                                                                 |                                                                                                               |
| Parameter: | rastport                                                                                                                                   | -> Zeiger auf die RastPort-Structure des Screens oder Windows, in den das Polygon gezeichnet werden soll.     |
|            | x,y                                                                                                                                        | -> Koordinaten des Eckpunktes.                                                                                |
| Ergebnis:  | fehler                                                                                                                                     | -> 0 bedeutet kein Fehler, -1 bedeutet, daß nicht genügend Speicherplatz für die Area-Liste reserviert wurde. |
| Datentyp:  | struct RastPort *rastport;<br>int x, y;<br>int fehler;                                                                                     |                                                                                                               |
| Sonstiges: | Das Polygon kann so viele Eckpunkte haben, wie benötigt werden. Allerdings muß genügend Speicherplatz mit InitArea reserviert worden sein. |                                                                                                               |
| Referenz:  | Für Areas siehe InitArea                                                                                                                   |                                                                                                               |

### 4.1.3 AreaEllipse

|           |                                                     |  |
|-----------|-----------------------------------------------------|--|
| Syntax:   | fehler = AreaEllipse(rastport,x,y,hradius,vradius); |  |
| Funktion: | Hängt eine Ellipse-Information an die Area-Liste an |  |

|            |                                                                                                                                                          |                                                                                                                               |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| Parameter: | rastport                                                                                                                                                 | -> Zeiger auf den RastPort des Screens oder Windows, in den die Ellipse gezeichnet werden soll.                               |
|            | x,y                                                                                                                                                      | -> Koordinaten des Mittelpunktes.                                                                                             |
|            | hradius                                                                                                                                                  | -> Radius auf der Horizontalen.                                                                                               |
|            | vradius                                                                                                                                                  | -> Radius auf den Vertikalen.                                                                                                 |
| Ergebnis:  | fehler                                                                                                                                                   | -> 0 bedeutet keine Probleme, 1 bedeutet, daß kein Speicherplatz für die Ellipse-Information in der Area-Liste vorhanden ist. |
| Datentyp:  | struct RastPort *rastport;<br>int x, y, hradius, vradius;<br>int fehler;                                                                                 |                                                                                                                               |
| Sonstiges: | Die Ellipse wird beim Ausführen der Area-Informationen ausgefüllt. Die Farbe, in der das geschieht, ist allgemein für die Areas zuvor festgelegt worden. |                                                                                                                               |
| Referenz:  | Für Areas siehe InitArea                                                                                                                                 |                                                                                                                               |

#### 4.1.4 **AreaEnd**

|            |                                                                                                                                                               |                                                                                    |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|
| Syntax:    | AreaEnd(rastport);                                                                                                                                            |                                                                                    |
| Funktion:  | Zeichnet die Kreise, Ellipsen und Polygone, die mit AreaCircle, AreaEllipse und AreaDraw deklariert worden sind.                                              |                                                                                    |
| Parameter: | rastport                                                                                                                                                      | -> Zeiger auf den RastPort, für den die Area-Informationen deklariert worden sind. |
| Ergebnis:  | Kein Ergebnis.                                                                                                                                                |                                                                                    |
| Datentyp:  | struct RastPort *rastport;                                                                                                                                    |                                                                                    |
| Sonstiges: | Dieser Befehl schließt die Area-Deklaration ab und zeichnet alles in den RastPort. Anschließend löscht er die Area-Liste, so daß sie neu benutzt werden kann. |                                                                                    |
| Referenz:  | Für Areas siehe InitArea                                                                                                                                      |                                                                                    |

## 4.1.5 AreaMove

|            |                                                                                                                                                              |                                                                                   |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|
| Syntax:    | fehler = AreaMove(rastport,x,y);                                                                                                                             |                                                                                   |
| Funktion:  | Schließt das vorhergehende Polygon, das mit einer Anzahl AreaDraw-Befehlen deklariert worden ist, ab und definiert einen Startpunkt für das nächste Polygon. |                                                                                   |
| Parameter: | rastport                                                                                                                                                     | -> Zeiger auf den rastPort des Screens oder Windows, für den die Area-Liste gilt. |
|            | x,y                                                                                                                                                          | -> Koordinaten des Startpunktes für das neue Polygon.                             |
| Ergebnis:  | fehler                                                                                                                                                       | -> 0, falls erfolgreich, 1, falls nicht.                                          |
| Datentyp:  | struct RastPort *rastport;<br>int x, y;<br>int fehler;                                                                                                       |                                                                                   |
| Sonstiges: | Dieser Befehl braucht nicht benutzt zu werden, wenn die Area-Liste abgeschlossen wird.                                                                       |                                                                                   |
| Referenz:  | Für Areas siehe InitArea                                                                                                                                     |                                                                                   |

## 4.1.6 BNDRYOFF

|            |                                                                                                                                                                                                                                   |                                                                                                          |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| Syntax:    | BNDRYOFF(rastport);                                                                                                                                                                                                               |                                                                                                          |
| Funktion:  | Schaltet die Randüberschreitungs-Kontrolle bei Area-End und RectFill ab                                                                                                                                                           |                                                                                                          |
| Parameter: | rastport                                                                                                                                                                                                                          | -> Zeiger auf die RastPort-Structure des Screens oder Windows, für die dieser Befehl Wirkung haben soll. |
| Ergebnis:  | Kein Ergebnis.                                                                                                                                                                                                                    |                                                                                                          |
| Datentyp:  | struct RastPort *rastport;                                                                                                                                                                                                        |                                                                                                          |
| Sonstiges: | Die Randüberschreitungskontrolle kontrolliert, ob beim Zeichnen mit RectFill oder bei den Area-Befehlen der Ausschnitt, der vom RastPort beschrieben wird, überschritten wurde. Im Normalfall wird dieser Befehl nicht verwendet. |                                                                                                          |

### 4.1.7 Draw

- Syntax:** Draw(rastport,x,y);
- Funktion:** Zeichnet von der momentanen Stiftposition zur angegebenen Position.
- Parameter:** rastport -> Zeiger auf die RastPort-Structure des RastPorts, in den gezeichnet werden soll.  
x,y -> Koordinaten der neuen Stiftposition.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct RastPort \*rastport;  
int x, y;
- Sonstiges:** Dieser Befehl zeichnet in der Farbe, die mit SetAPen, SetBPen, SetOPen und SetDrMd eingestellt werden kann.

### 4.1.8 DrawCircle

- Syntax:** DrawCircle(rastport,x,y,radius);
- Funktion:** Zeichnet einen Kreis.
- Parameter:** rastport -> Zeiger auf die RastPort-Structure des RastPorts, in den gezeichnet werden soll.  
x,y -> Koordinaten des Mittelpunktes.  
radius -> Radius des Kreises in Pixel.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct RastPort \*rastport;  
int x, y, radius;
- Sonstiges:** Dieser Befehl zeichnet in der Farbe, die mit SetAPen, SetBPen, SetOPen und SetDrMd eingestellt werden kann.

## 4.1.9 DrawEllipse

|            |                                                                                                                                                                   |                                                                                    |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|
| Syntax:    | <code>DrawEllipse(rastport,x,y,hradius,vradius);</code>                                                                                                           |                                                                                    |
| Funktion:  | Zeichnet eine Ellipse                                                                                                                                             |                                                                                    |
| Parameter: | <code>rastport</code>                                                                                                                                             | -> Zeiger auf die RastPort-Structure des RastPorts, in den gezeichnet werden soll. |
|            | <code>x,y</code>                                                                                                                                                  | -> Koordinaten des Mittelpunktes.                                                  |
|            | <code>hradius</code>                                                                                                                                              | -> Radius auf der horizontalen Achse.                                              |
|            | <code>vradius</code>                                                                                                                                              | -> Radius auf der vertikalen Achse.                                                |
| Ergebnis:  | Kein Ergebnis.                                                                                                                                                    |                                                                                    |
| Datentyp:  | <code>struct RastPort *rastport;</code><br><code>int x, y, hradius, vradius;</code>                                                                               |                                                                                    |
| Sonstiges: | Dieser Befehl zeichnet in der Farbe, die mit <code>SetAPen</code> , <code>SetBPen</code> , <code>SetOPen</code> und <code>SetDrMd</code> eingestellt werden kann. |                                                                                    |

## 4.1.10 Flood

|            |                                                                                                                                                                                                                                                                                                                              |                                                                                                |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| Syntax:    | <code>Flood(rastport,mode,x,y);</code>                                                                                                                                                                                                                                                                                       |                                                                                                |
| Funktion:  | Füllt einen Bereich aus.                                                                                                                                                                                                                                                                                                     |                                                                                                |
| Parameter: | <code>rastport</code>                                                                                                                                                                                                                                                                                                        | -> Zeiger auf die RastPort-Structure des RastPorts, in dem ein Bereich ausgefüllt werden soll. |
|            | <code>mode</code>                                                                                                                                                                                                                                                                                                            | -> gibt den Füllmodus an.                                                                      |
|            | <code>x,y</code>                                                                                                                                                                                                                                                                                                             | -> Position, von der aus gefüllt werden soll.                                                  |
| Ergebnis:  | Kein Ergebnis.                                                                                                                                                                                                                                                                                                               |                                                                                                |
| Datentyp:  | <code>struct RastPort *rastport;</code><br><code>int mode, x, y;</code>                                                                                                                                                                                                                                                      |                                                                                                |
| Sonstiges: | Ist <code>mode</code> gleich 1, wird der Bereich ausgefüllt, der die gleiche Farbe hat, wie der Punkt mit den Koordinaten <code>x</code> und <code>y</code> .<br><br>Ist <code>mode</code> gleich 0, wird soweit gefüllt, bis die Funktion auf Punkte mit der Farbe trifft, die mit <code>SetOPen</code> gesetzt worden ist. |                                                                                                |

Um Flood verwenden zu können, muß zuerst mit AllocRaster und InitTmpRas, genau wie bei Areas, ein Speicherbereich für die Verwendung durch Flood, bzw. Area reserviert werden.

### 4.1.11 GetRGB4

**Syntax:** farbwert = GetRGB4(CMap,Nr);

**Funktion:** Holt den Farbwert aus dem spezifizierten Farbregister

**Parameter:** CMap -> Zeiger auf die ColorMap.  
Nr -> Farbregisternummer.

**Ergebnis:** farbwert -> Farbwert des spezifizierten Registers.

**Datentyp:** struct ColorMap \*CMap;  
int Nr;  
WORD farbwert;

**Sonstiges:** Auf die ColorMap kann folgendermaßen zugegriffen werden:  
ScreenPtr->ViewPort->ColorMap  
oder aber vom Window-Pointer aus:  
WindowPtr->WScreen->ViewPort->ColorMap

Der Farbwert kann hexadezimal am einfachsten ausgewertet werden. Er ist wie folgt aufgebaut:  
0x0RGB (R=Rotwert; G=Grünwert; B=Blauwert)

### 4.1.12 InitArea

**Syntax:** InitArea(info,speicher,maxein);

**Funktion:** Initialisiert die Area-Liste.

**Parameter:** info -> Zeiger auf die AreaInfo-Structure des zugehörigen RastPorts.  
speicher -> Zeiger auf einen freien Speicherbereich für die Area-Informationen.  
maxein -> Die maximale Anzahl von Koordina-teneinträgen in die Liste.

**Ergebnis:** Kein Ergebnis.

Datentyp: struct AreaInfo \*info;  
 ULONG speicher;  
 int maxein;

Sonstiges: Auf die AreaInfo kann folgendermaßen zugegriffen werden:

Window:WindowPtr -> RPort->AreaInfo

Screen:ScreenPtr -> RastPort->AreaInfo

Areas sind spezielle Grafikteile des Amiga. Mit ihrer Hilfe können überaus leicht ganze Listen von Grafikelementen aufgebaut werden, die mittels AreaEnd auf einmal auf den Bildschirm gezeichnet werden. Es gibt drei Grafikelemente, die bei Areas verwendet werden können:

1. Kreise: Mittels AreaCircle können Kreise in die Area-Liste eingetragen werden, die beim Zeichnen ausgefüllt werden.

2. Ellipsen: Ellipsen sind im Prinzip genauso zu behandeln, wie Kreise, nur daß sie mit AreaEllipse eingetragen werden.

3. Polygone: Polygone sind die wohl am meisten verwendeten Grafikelemente bei Areas. Sie können aus beliebig vielen Eckpunkten bestehen, die mit AreaDraw in die Liste eingetragen werden müssen. Soll ein Polygon abgeschlossen werden und ein neues begonnen werden, so muß der Befehl AreaMove verwendet werden.

Als letztes wird dann der Befehl AreaEnd gegeben, der das Zeichnen der Grafikelemente veranlaßt. Anzumerken ist, daß alle Elemente ausgefüllt werden und zwar in der Farbe, die mit SetOPen gesetzt wird.

### 4.1.13 InitTmpRas

Syntax: InitTmpRas(tmpras,speicher,größe);

Funktion: Reserviert Speicherplatz für die Verwendung durch Areas.

Parameter: tmpras -> Zeiger auf die TmpRas-Structure.  
 speicher -> Ist der Zeiger, der durch diesen Befehl auf den Beginn des reservierten Speicherplatzes zeigt.  
 gröÙe -> gibt die Größe des zu reservierenden Speicherplatzes an.

Ergebnis: Kein Ergebnis.

**Datentyp:** struct TmpRas \*tmpras;  
ULONG speicher;  
int grÖÙe;

**Sonstiges:** Auf die TmpRas-Structure kann wie folgt zugegriffen werden:  
Window:WindowPtr -> RPort->TmpRas  
Screen:ScreenPtr -> RastPort->TmpRas  
Die Variable "speicher" wird für die Verwendung bei InitArea benötigt. "grÖÙe" sollte etwa fünfmal so groß sein wie "maxein" (siehe InitArea).

#### 4.1.14 Move

**Syntax:** Move(rastport,x,y);

**Funktion:** Plaziert den Stift an der spezifizierten Position neu.

**Parameter:** rastport -> Zeiger auf die RastPort-Structure des RastPorts, in dem der Stift neu plaziert werden soll.  
x,y -> Die neuen Koordinaten des Stiftes.

**Ergebnis:** Kein Ergebnis.

**Datentyp:** struct RastPort \*rastport;  
int x, y;

**Sonstiges:** Dieser Befehl hat etwa die gleiche Funktion wie Draw. Allerdings wird hier nicht gezeichnet, sondern nur verschoben.

#### 4.1.15 OFF\_DISPLAY

**Syntax:** OFF\_DISPLAY;

**Funktion:** Schaltet die Bildschirmdarstellung aus.

**Parameter:** Keine Parameter.

**Ergebnis:** Kein Ergebnis.

**Datentyp:** Keine Variablen.

**Sonstiges:** Es ist solange nichts zu sehen, bis ON\_DISPLAY verwendet wird.

OFF\_DISPLAY ist ein Makro und befindet sich in »gfxmacros.h«. Es muß also folgendermaßen eingelsen werden:

```
#include <graphics/gfxmacros.h>
```

Referenz: Siehe auch ON\_DISPLAY.

#### 4.1.16 ON\_DISPLAY

Syntax: ON\_DISPLAY;

Funktion: Schaltet die Bildschirmdarstellung an.

Parameter: Keine Parameter.

Ergebnis: Kein Ergebnis.

Datentyp: Keine Variablen.

Sonstiges: Dieser Zustand ist normal.

ON\_DISPLAY ist ein Macro und befindet sich in »gfxmacros.h«. Es muß also folgendermaßen eingelsen werden:

```
#include <graphics/gfxmacros.h>
```

Referenz: Siehe auch OFF\_DISPLAY

#### 4.1.17 PolyDraw

Syntax: PolyDraw(rastport,anz,&anfang[0]);

Funktion: Zeichnet ein Polygon.

Parameter: rastport -> Zeiger auf die RastPort-Structure des RastPorts, in den gezeichnet werden soll.

anz -> Anzahl der Eckpunkte, die das Polygon besitzt.

&anfang[0] -> Zeiger auf das erste xy-Koordinaten-Paar.

Ergebnis: Kein Ergebnis.

Datentyp: struct RastPort \*rastport;  
int anz;  
USHORT anfang[];

Sonstiges: Wie dieser Befehl genau angewendet wird, ist aus dem Demonstrationsprogramm ersichtlich.

### 4.1.18 ReadPixel

Syntax: `Nr = ReadPixel(rastport,x,y);`

Funktion: Ermittelt die Farbregisternummer des spezifizierten Punktes.

Parameter: `rastport` -> Zeiger auf die RastPort-Structure des Rastportes, auf die sich dieser Befehl bezieht.

`x,y` -> Koordinaten des Punktes, von dem die Farbregisternummer ermittelt werden soll.

Ergebnis: `Nr` -> Farbregisternummer des spezifizierten Pixel.

Datentyp: `struct RastPort *rastport;`  
`int x, y;`  
`int Nr;`

Sonstiges: Wenn `Nr` gleich `-1` ist, konnte die Farbregisternummer des Pixel nicht ermittelt werden. `Nr` kann die Werte `0` bis `255` und `-1` annehmen. Dies wurde so gewählt, da eine spätere Version des Amiga eventuell mit 8 Bitplanes arbeitet, also 256 Farben gleichzeitig darstellen kann.

Erwähnenswert ist noch, daß die Farbregisternummer zurückgegeben wird, nicht der Farbwert selbst. Dieser kann anschließend mit `GetRGB4` ermittelt werden.

### 4.1.19 RectFill

Syntax: `RectFill(rastport,xl,yo,xr,yu);`

Funktion: Erzeugt ein ausgefülltes Rechteck.

Parameter: `rastport` -> Zeiger auf die RastPort-Structure des RastPorts, in den gezeichnet werden soll.

`xl,yo` -> Koordinaten des linken, oberen Eckpunktes

`xr,yu` -> Koordinaten des rechten, unteren Eckpunktes.

|            |                                                                                                                                           |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| Ergebnis:  | Kein Ergebnis.                                                                                                                            |
| Datentyp:  | struct RastPort *rastport;<br>int xl, yo, xr, yu;                                                                                         |
| Sonstiges: | Gezeichnet wird in der gewählten Zeichenfarbe und mit dem gewählten Füllmuster. Ist kein Füllmuster gewählt, wird vollständig ausgefüllt. |

#### 4.1.20 ScrollRaster

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                              |    |                                                                                |       |    |                                                                                                                              |       |    |                                                                   |       |    |                                                                     |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|----|--------------------------------------------------------------------------------|-------|----|------------------------------------------------------------------------------------------------------------------------------|-------|----|-------------------------------------------------------------------|-------|----|---------------------------------------------------------------------|
| Syntax:    | ScrollRaster(rastport,dx,dy,xl,yo,xr,yu);                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                                                                                              |    |                                                                                |       |    |                                                                                                                              |       |    |                                                                   |       |    |                                                                     |
| Funktion:  | Scrollt einen gewählten Bereich.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                              |    |                                                                                |       |    |                                                                                                                              |       |    |                                                                   |       |    |                                                                     |
| Parameter: | <table> <tr> <td>rastport</td> <td>-&gt;</td> <td>Zeiger auf die RastPort-Structure des RastPorts, in dem gescrollt werden soll.</td> </tr> <tr> <td>dx,dy</td> <td>-&gt;</td> <td>geben an, um wieviele Punkte gescrollt werden soll. Wird ein negativer Wert angegeben, wird nach links, bzw. oben gescrollt.</td> </tr> <tr> <td>xl,yo</td> <td>-&gt;</td> <td>Linker, oberer Eckpunkt des Bereiches, der gescrollt werden soll.</td> </tr> <tr> <td>xr,yu</td> <td>-&gt;</td> <td>Rechter, unterer Eckpunkt des Bereiches, der gescrollt werden soll.</td> </tr> </table> | rastport                                                                                                                     | -> | Zeiger auf die RastPort-Structure des RastPorts, in dem gescrollt werden soll. | dx,dy | -> | geben an, um wieviele Punkte gescrollt werden soll. Wird ein negativer Wert angegeben, wird nach links, bzw. oben gescrollt. | xl,yo | -> | Linker, oberer Eckpunkt des Bereiches, der gescrollt werden soll. | xr,yu | -> | Rechter, unterer Eckpunkt des Bereiches, der gescrollt werden soll. |
| rastport   | ->                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Zeiger auf die RastPort-Structure des RastPorts, in dem gescrollt werden soll.                                               |    |                                                                                |       |    |                                                                                                                              |       |    |                                                                   |       |    |                                                                     |
| dx,dy      | ->                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | geben an, um wieviele Punkte gescrollt werden soll. Wird ein negativer Wert angegeben, wird nach links, bzw. oben gescrollt. |    |                                                                                |       |    |                                                                                                                              |       |    |                                                                   |       |    |                                                                     |
| xl,yo      | ->                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Linker, oberer Eckpunkt des Bereiches, der gescrollt werden soll.                                                            |    |                                                                                |       |    |                                                                                                                              |       |    |                                                                   |       |    |                                                                     |
| xr,yu      | ->                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Rechter, unterer Eckpunkt des Bereiches, der gescrollt werden soll.                                                          |    |                                                                                |       |    |                                                                                                                              |       |    |                                                                   |       |    |                                                                     |
| Ergebnis:  | Kein Ergebnis.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                                                                                                              |    |                                                                                |       |    |                                                                                                                              |       |    |                                                                   |       |    |                                                                     |
| Datentyp:  | struct RastPort *rastport;<br>int dx, dy, xl, yo, xr, yu;                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                                                                                              |    |                                                                                |       |    |                                                                                                                              |       |    |                                                                   |       |    |                                                                     |
| Sonstiges: | Der freiwerdende Bereich wird mit der Farbe gefüllt, die mit SetBPen gewählt werden kann.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                                                                                              |    |                                                                                |       |    |                                                                                                                              |       |    |                                                                   |       |    |                                                                     |

#### 4.1.21 ScrollVPort

|            |                                                                                                                                                                                                                                                                           |                                                                           |    |                                                                           |       |    |                                                                         |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|----|---------------------------------------------------------------------------|-------|----|-------------------------------------------------------------------------|
| Syntax:    | ScrollVPort(viewport,dx,dy);                                                                                                                                                                                                                                              |                                                                           |    |                                                                           |       |    |                                                                         |
| Funktion:  | Scrollt einen ganzen Screenbereich.                                                                                                                                                                                                                                       |                                                                           |    |                                                                           |       |    |                                                                         |
| Parameter: | <table> <tr> <td>viewport</td> <td>-&gt;</td> <td>Zeiger auf die ViewPort-Structure des Screens, der gescrollt werden soll.</td> </tr> <tr> <td>dx,dy</td> <td>-&gt;</td> <td>Anzahl der Punkte um die gescrollt werden soll. Wird ein negativer Wert</td> </tr> </table> | viewport                                                                  | -> | Zeiger auf die ViewPort-Structure des Screens, der gescrollt werden soll. | dx,dy | -> | Anzahl der Punkte um die gescrollt werden soll. Wird ein negativer Wert |
| viewport   | ->                                                                                                                                                                                                                                                                        | Zeiger auf die ViewPort-Structure des Screens, der gescrollt werden soll. |    |                                                                           |       |    |                                                                         |
| dx,dy      | ->                                                                                                                                                                                                                                                                        | Anzahl der Punkte um die gescrollt werden soll. Wird ein negativer Wert   |    |                                                                           |       |    |                                                                         |

angegeben, wird nach links bzw. oben gescrollt.

Ergebnis: Kein Ergebnis.

Datentyp: struct ViewPort \*viewport;  
int dx, dy;

Sonstiges: Dieses Scrolling geschieht zwar langsamer als beim MoveScreen-Befehl, vollzieht sich aber immerhin in nur etwa 1/60 Sekunde.

Der freiwerdende Bereich wird mit der Farbe gefüllt, die mit SetBPen gesetzt werden kann.

### 4.1.22 SetAfPt

Syntax: SetAfPt(rastport,&data[0],anz);

Funktion: Setzt ein Füllmuster, das bei Area-Befehlen verwendet wird.

Parameter: rastport -> Zeiger auf die RastPort-Structure des RastPorts, für den dieses Füllmuster gelten soll.

&data[0] -> Zeiger auf ein Feld, das das Füllmuster enthält.

anz -> 2 hoch anz ist gleich der Anzahl der Worte, die data enthält.

Ergebnis: Kein Ergebnis.

Datentyp: struct RastPort \*rastport;  
WORD data[];  
int anz;

Sonstiges: »data« muß immer quadratisch sein. Das heißt, es muß genauso viele Worte breit wie hoch sein. Da anz immer als Wurzel aus der Anzahl der Worte angegeben wird, können insgesamt entweder 1, 2, 4, 8, 16 usw. Worte verwendet werden.

Wir wollen nun ein Linienmuster mit jeweils 2 Worten als Kantenlänge definieren:

```
1010101010101010
0101010101010101
```

Daraus folgt folgende Deklaration:

```
.
WORD data [] = {0xAAAA 0x5555};
.br/>main()
{
.br/>SetAfPt (WindowPtr->RPort, &data[0], 1);
.br/>}
```

### 4.1.23 SetAPen

- Syntax:** SetAPen(rastport,Nr);
- Funktion:** Setzt die Zeichenfarbe.
- Parameter:** rastport -> Zeiger auf die RastPort-Structure des RastPorts, für den die Zeichenfarbe gesetzt werden soll.
- Nr -> Farbregister.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct RastPort \*rastport;  
int Nr;
- Sonstiges:** Da beim Amiga zum Zeichnen nicht die Farbe direkt angegeben wird, sondern über Farbregister, muß hier eine Farbregisternummer angegeben werden.
- Die Zeichenfarbe wird für alle direkten Zeichenbefehle verwendet, wie zum Beispiel Draw usw.

### 4.1.24 SetBPen

- Syntax:** SetBPen(rastport,Nr);
- Funktion:** Setzt die Hintergrundfarbe.
- Parameter:** rastport -> Zeiger auf die RastPort-Structure des RastPorts, dessen Hintergrundfarbe gesetzt werden soll.
- Nr -> Farbregisternummer für die Hintergrundfarbe.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct RastPort \*rastport;  
int Nr;

**Sonstiges:** Hintergrundfarbe ist hier nicht als die Farbe zu verstehen, die die Umrandung des Bildschirms besitzt, sondern als Farbe, die zum Beispiel zum Füllen des freiwerdenden Bereiches beim Scrollen verwendet wird.

### 4.1.25 SetDrMd

**Syntax:** - SetDrMd(rastport,mode);

**Funktion:** Setzt den Mode, mit dem gezeichnet wird.

**Parameter:** rastport -> Zeiger auf die RastPort-Structure des RastPorts, dessen Zeichenmode gesetzt werden soll.

mode -> JAM1 - Normalmodus, d.h. die Farbe, die mit SetAPen gesetzt wurde, wird zum Zeichnen verwendet.

JAM2 - wird für Füllmuster verwendet. Ist bei einem Füllmuster ein "Loch", d.h. ist das Bit nicht gesetzt, wird die Hintergrundfarbe (SetBPen) verwendet. Ansonsten wird in der Vordergrundfarbe (SetAPen) gezeichnet.

COMPLEMENT|JAM1 - komplementiert die Vordergrundfarbe.

COMPLEMENT|JAM2 - komplementiert entweder die Vorder- oder die Hintergrundfarbe (siehe JAM2).

JAM1|INVERSVID, beziehungsweise JAM2|INVERSVID wird für die Textdarstellung benutzt. In diesem Fall wird die Textdarstellung invertiert.

**Ergebnis:** Kein Ergebnis.

**Datentyp:** struct RastPort \*rastport;  
int mode;

**Sonstiges:** Im Normalfall wird der Modus JAM1 verwendet:

SetDrMd(WindowPtr->RPort,JAM1);

### 4.1.26 SetDrPt

|            |                                                                                                                                                         |                                                                                              |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| Syntax:    | SetDrPt(rastport,data);                                                                                                                                 |                                                                                              |
| Funktion:  | Setzt ein Muster für die Linienzeichnung.                                                                                                               |                                                                                              |
| Parameter: | rastport                                                                                                                                                | -> Zeiger auf die RastPort-Structure des RastPorts, dessen Linienmuster gesetzt werden soll. |
|            | data                                                                                                                                                    | -> ist eine Wortvariable, die das Muster enthält.                                            |
| Ergebnis:  | Kein Ergebnis.                                                                                                                                          |                                                                                              |
| Datentyp:  | struct RastPort *rastport;<br>WORD data;                                                                                                                |                                                                                              |
| Sonstiges: | Wir wollen folgendes Muster setzen: 1010101010101010<br>Dies ist gleich 0xAAAA, also bedeutet dies :<br><pre>SetDrPt(WindowPtr-&gt;RPort,0xAAAA);</pre> |                                                                                              |

### 4.1.27 SetOPen

|            |                                                                                         |                                                                                              |
|------------|-----------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| Syntax:    | SetOPen(rastport,Nr);                                                                   |                                                                                              |
| Funktion:  | Setzt die Farbe, die von einigen Befehlen verwendet wird (Beispielsweise ScrollRaster). |                                                                                              |
| Parameter: | rastport                                                                                | -> Zeiger auf die RastPort-Structure des RastPorts, für den diese Farbe gesetzt werden soll. |
|            | Nr                                                                                      | -> Farbregisternummer.                                                                       |
| Ergebnis:  | Kein Ergebnis.                                                                          |                                                                                              |
| Datentyp:  | struct RastPort *rastport;<br>int Nr;                                                   |                                                                                              |
| Sonstiges: | Für weitere Informationen über die inverse Textdarstellung siehe auch SetDrMd.          |                                                                                              |

### 4.1.28 SetRast

|           |                                                                                     |  |
|-----------|-------------------------------------------------------------------------------------|--|
| Syntax:   | SetRast(rastport,Nr);                                                               |  |
| Funktion: | Färbt ein gesamtes Window, bzw. einen gesamten Screen in der gewünschten Farbe ein. |  |

|            |                                                                                             |                                                                                 |
|------------|---------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------|
| Parameter: | rastport                                                                                    | -> Zeiger auf die RastPort-Structure des RastPorts, der eingefärbt werden soll. |
|            | Nr                                                                                          | -> Farbregisternummer, das die Farbe angibt, in der eingefärbt werden soll.     |
| Ergebnis:  | Kein Ergebnis.                                                                              |                                                                                 |
| Datentyp:  | struct RastPort *rastport;<br>int Nr;                                                       |                                                                                 |
| Sonstiges: | Die ist der einzige Befehl, der es erlaubt, einen gesamten RastPort auf einmal einzufärben. |                                                                                 |

### 4.1.29 SetRGB4

|            |                                                                                                      |                                                                                                    |
|------------|------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|
| Syntax:    | SetRGB4(viewport,Nr,r,g,b);                                                                          |                                                                                                    |
| Funktion:  | Setzt ein Farbregister auf den spezifizierten Farbwert                                               |                                                                                                    |
| Parameter: | viewport                                                                                             | -> Zeiger auf die ViewPort-Structure des Screens, für den die Farbe gesetzt werden soll.           |
|            | Nr                                                                                                   | -> Nummer des Farbregisters, das gesetzt werden soll.                                              |
|            | r,g,b                                                                                                | -> geben die Intensität des Rot/Grün/Blau-Anteils an. Sie dürfen Werte zwischen 0 und 15 annehmen. |
| Ergebnis:  | Kein Ergebnis.                                                                                       |                                                                                                    |
| Datentyp:  | struct ViewPort *viewport;<br>int Nr, r, g, b;                                                       |                                                                                                    |
| Sonstiges: | Werden r, g und b gleich 0 gewählt, bedeutet dies schwarz. Für r, g und b gleich 15 erhält man weiß. |                                                                                                    |

### 4.1.30 SetWrMsk

|            |                                                    |                                                                                               |
|------------|----------------------------------------------------|-----------------------------------------------------------------------------------------------|
| Syntax:    | SetWrMsk(rastport,maske);                          |                                                                                               |
| Funktion:  | Schützt bestimmte Bitplanes vor dem Überschreiben. |                                                                                               |
| Parameter: | rastport                                           | -> Zeiger auf die RastPort-Structure des RastPorts, dessen BitPlanes geschützt werden sollen. |

maske -> gibt an, welche BitPlanes vor dem Überschreiben geschützt werden sollen.

Ergebnis: Kein Ergebnis.

Datentyp: struct RastPort \*rastport;  
WORD maske;

Sonstiges: maske ist vom Typ WORD, also 16 Bits lang. Die unteren 6 Bits beschreiben den Schreib-Zustand der maximal 6 BitPlanes, die ein Screen haben kann.

Wir wollen nun die Planes 2 und 4 vor dem Überschreiben schützen:

BitPlane 6543210  
111111111101011 = 0xFFEB

Man verwendet also folgenden Befehl:

SetWrMsk(WindowPtr->RPort,0xFFEB);

### 4.1.31 VBeamPos

Syntax: YPos = VBeamPos();

Funktion: Ermittelt die momentane Y-Position des Elektronenstrahles in der Bildröhre.

Parameter: Keine Parameter.

Ergebnis: YPos -> die momentane Position in non-interlaced Mode.

Datentyp: int YPos;

Sonstiges: Dieser Befehl fragt direkt die Hardware ab.

### 4.1.32 WaitBOVP

Syntax: WaitBOVP(viewport);

Funktion: Wartet, bis der Elektronenstrahl den Anfang des angegebenen Screens erreicht hat.

Parameter: viewport -> Zeiger auf die ViewPort-Structure eines Screens.

Ergebnis: Kein Ergebnis.

- Datentyp:** struct ViewPort \*viewport;
- Sonstiges:** Dieser Befehl kann verwendet werden, wenn das Programm direkt auf die Video-Hardware zugreift und mit dieser in Synchronität gebracht werden muß.
- Das Programm wird weiter abgearbeitet, sobald der Elektronenstrahl mit der Darstellung des angegebenen Screens beginnt. Dies muß nicht unbedingt die oberste Position des Bildschirms sein, da der Screen auch weiter unten beginnen kann (MoveScreen).

### 4.1.33 WaitTOF

- Syntax:** WaitTOF();
- Funktion:** Wartet, bis der Elektronenstrahl den Beginn des nächsten, darzustellenden Screens erreicht hat.
- Parameter:** Keine Parameter.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** Keine Variablen.
- Sonstiges:** Auch dieser Befehl kann zur Synchronisation des Programms mit der Hardware herangezogen werden.

### 4.1.34 WritePixel

- Syntax:** WritePixel(rastport,x,y);
- Funktion:** Dieser Befehl setzt einen einzelnen Punkt, in der mit SetAPen angegebenen Farbe an die angegebene Position.
- Parameter:** rastport -> Zeiger auf die RastPort-Structure des RastPorts, in die der Punkt gesetzt werden soll.
- x,y -> Koordinaten des Punktes.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct RastPort \*rastport;  
int x, y;
- Sonstiges:** Dieser Befehl kann auch durch Draw ersetzt werden, wenn der Zeichenstift zuvor mit Move an die Punktposition gesetzt wurde.

```

1 /*****
2
3 Grafik-Demonstration
4 last update 26/05/87
5 von Joerg Koch und Frank Kremser
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Diese Demonstration zeigt die einfachen Grafikbefehle des Amiga
11
12 *****/
13
14 #include <exec/types.h> /* Laedt die Include-Files */
15 #include <exec/nodes.h>
16 #include <exec/lists.h>
17 #include <exec/ports.h>
18 #include <exec/devices.h>
19 #include <devices/keymap.h>
20 #include <graphics/regions.h>
21 #include <graphics/copper.h>
22 #include <graphics/gels.h>
23 #include <graphics/gfxbase.h>
24 #include <graphics/gfx.h>
25 #include <graphics/clip.h>
26 #include <graphics/view.h>
27 #include <graphics/rastport.h>
28 #include <graphics/layers.h>
29 #include <graphics/display.h>
30 #include <graphics/gfxmacros.h>
31 #include <intuition/intuition.h>
32 #include <hardware/blit.h>
33
34 struct IntuitionBase *IntuitionBase; /* Lib - Zeiger */
35 struct GfxBase *GfxBase;
36
37 struct RastPort *rp; /* RastPort - Zeiger */
38 struct Window *window; /* Window - Zeiger */
39 struct IntuiMessage *message; /* IntuiMessage - Zeiger */
40 struct TmpRas tmp;
41
42 struct NewWindow nw = /* Window definieren */
43 {
44 30, /* Linke Ecke */
45 30, /* Rechte Ecke */
46 580, /* Breite */
47 200, /* Hoehe */
48 3, /* DetailPen */
49 1, /* Block Pen */
50 CLOSEWINDOW|REFRESHWINDOW, /* IDCMP-Flags */
51 WINDOWCLOSE|WINDOWSIZING|WINDOWDEPTH|WINDOWDRAG|SMART_REFRESH, /* Flags */
52 NULL, /* Erstes Gadget des Windows */
53 NULL, /* Checkmark */
54 "Die einfachen Grafikbefehle", /* Window-Titel */
55 NULL, /* Zeiger auf Screen */
56 NULL, /* Zeiger auf SuperBitMap */
57 20, /* Min. Breite */
58 20, /* Min. Hoehe */
59 640, /* Max. Breite */
60 256, /* Max. Hoehe */
61 WBENCHSCREEN /* Screen-Typ */
62 };
63
64 LONG mem;

```

```

65
66
67 main()
68 {
69 ULONG MessageClass;
70 USHORT code;
71 LONG warte;
72
73 /* Grafik-Bibliothek oeffnen */
74 if(!(GfxBase = (struct GfxBase *)OpenLibrary("graphics.library",0)))
75 {
76 close_things();
77 exit();
78 }
79 /* Intuition oeffnen */
80 if(!(IntuitionBase = (struct IntuitionBase *)
81 OpenLibrary("intuition.library",0)))
82 {
83 close_things();
84 exit();
85 }
86 if (!(window = (struct Window *)OpenWindow(&nw)) /* Window oeffnen */
87 {
88 close_things();
89 exit();
90 }
91
92 rp = window->RPort;
93 mem = AllocRaster(640,256); /* Speicherplatz fuer Flood bereitstellen */
94 rp->TmpRas = (struct TmpRas *)InitTmpRas(&tmp
95 ,mem, RASSIZE(640,256));
96
97 SetDrMd(rp,JAM1); /* Drawmode setzen */
98 SetAPen(rp,2); /* Farbregerister zum Zeichnen setzen */
99 Move(rp,20,20); /* Stift neu positionieren */
100 Draw(rp,20,180); /* Linie zur neuen Position zeichnen */
101 Draw(rp,560,180);
102 Draw(rp,560,20);
103 Draw(rp,20,20);
104 Draw(rp,560,180);
105 Move(rp,560, 20);
106 Draw(rp,20,180);
107
108 /* Flaechen fuehlen */
109 Flood(rp,1,30,22);
110 Flood(rp,1,30,178);
111
112 for(warte=0; warte<100000; warte++); /* Einen Moment warten */
113
114 SetBPen(rp,0); /* Hintergrundfarbe setzen und scrollen */
115 for(warte=0; warte<160; warte++) ScrollRaster(rp,4,1,10,10,570,190);
116
117 for(warte=0; warte<50000; warte++); /* Einen Moment warten */
118
119 RectFill(rp,1,10,578,198); /* ausgefuelltes Rechteck */
120
121 SetAPen(rp,3);
122 for(warte=0; warte<580; warte=warte+3) WritePixel(rp,warte,100);
123 /* gestrichelte Linie zeichnen */
124 for(;;) /* Endlosschleife */
125 {
126 /* Message empfangen und verarbeiten */
127 if (message = (struct IntuiMessage *)GetMsg(window->UserPort))
128 {
129 MessageClass = message->Class; /* Message retten */
130 code = message->Code;

```

```

129 ReplyMsg(message); /* Message quittieren */
130 if (MessageClass == CLOSEWINDOW) {close_things(); exit();};
131 }; /* Wenn bestimmte Message, dann tue... */
132 }
133 }
134
135 close_things() /* Unterroutine zum Abschluss des Programm */
136 {
137 FreeRaster(mem,640,256);
138 CloseWindow(window);
139 CloseLibrary(GfxBase);
140 CloseLibrary(IntuitionBase);
141 }

1 /*****
2
3 Area - Demonstration
4 last update 25/05/87
5 von Frank Kremser und Joerg Koch
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Zeichnet zwei Polygone auf den Screen
11
12 *****/
13
14 #include <exec/types.h> /* Include - Files einlesen */
15 #include <exec/nodes.h>
16 #include <exec/lists.h>
17 #include <exec/ports.h>
18 #include <exec/devices.h>
19 #include <devices/keymap.h>
20 #include <graphics/regions.h>
21 #include <graphics/copper.h>
22 #include <graphics/gels.h>
23 #include <graphics/gfxbase.h>
24 #include <graphics/gfx.h>
25 #include <graphics/clip.h>
26 #include <graphics/view.h>
27 #include <graphics/rastport.h>
28 #include <graphics/layers.h>
29 #include <graphics/display.h>
30 #include <graphics/gfxmacros.h>
31 #include <intuition/intuition.h>
32 #include <hardware/blit.h>
33
34 struct IntuitionBase *IntuitionBase; /* Intuition-Pointer definieren */
35 struct GfxBase *GfxBase; /* Graphics-Lib-Pointer definieren */
36 struct Screen *screen; /* Screen - Pointer definieren */
37
38 struct NewScreen ns = /* Screen - Struktur */
39 {
40 0,
41 0, /* Startposition */
42 640, /* Breite */
43 256, /* Hoehe */
44 2, /* Tiefe */
45 1, /* DetailPen */
46 0, /* BlockPen */
47 HIRES, /* ViewModes */

```

```

48 CUSTOMSCREEN, /* Screen-Typ */
49 NULL, /* Zeiger auf Font - Structure */
50 "Die Area-Befehle", /* Screen - Titel */
51 NULL, /* Gadgets */
52 NULL, /* selbsterstelltes BitMap */
53 };
54
55
56 main()
57 {
58 LONG warte; /* Variable fuer Warte - Schleife */
59 WORD areabuffer[250]; /* areapuffer = 250 Words gross */
60 struct RastPort *rp; /* rp = Pointer auf RastPort */
61 struct TmpRas tmp; /* tmp = TmpRas - Structure */
62 struct AreaInfo areainfo; /* areainfo = AreaInfo - Structure */
63 LONG mem; /* Variable fuer Speicherplatz */
64 /* Libs oeffnen */
65 if(!(GfxBase = (struct GfxBase *)
66 OpenLibrary("graphics.library",0))) exit();
67
68 if(!(IntuitionBase = (struct IntuitionBase *)
69 OpenLibrary("intuition.library",0))) exit();
70
71 if (!(screen = (struct Screen *)OpenScreen(&ns))) exit(); /* Screen oeffnen */
72
73 rp = &screen->RastPort;
74 mem = AllocRaster(640,256); /* Speicher fuer Areas bereitstellen */
75 /* Area initialisieren */
76 InitArea(&areainfo, areabuffer, 100);
77 rp->TmpRas = (struct TmpRas *)InitTmpRas(&tmp
78 ,mem,RASSIZE(640,256));
79 rp->AreaInfo = &areainfo;
80
81 SetDrMd(rp,JAM1); /* Zeichenmodus und Farbe setzen */
82 SetAPen(rp,3);
83
84 AreaMove(rp,0,0); /* Polygone definieren */
85 AreaDraw(rp,30,50);
86 AreaDraw(rp,100,100);
87 AreaDraw(rp,200,50);
88 AreaDraw(rp,100,250);
89
90 AreaMove(rp,400,100);
91 AreaDraw(rp,450,100);
92 AreaDraw(rp,450,150);
93 AreaDraw(rp,400,150);
94
95 AreaEnd(rp); /* Polygone zeichnen */
96
97 for(warte = 0; warte < 300000; warte++); /* Warte-Schleife */
98
99 FreeRaster(mem,640,256); /* Area-Speicher loeschen */
100 CloseScreen(screen); /* Screen und Libs schliessen */
101 CloseLibrary(GfxBase);
102 CloseLibrary(IntuitionBase);
103 }

```

## 4.2 Die Textfunktionen

Der Amiga stellt nicht nur mächtige Grafikbefehle zur Verfügung, sondern er beherrscht auch die Fähigkeit der Schriftenvielfalt. Mit einigen Befehlen kann sehr leicht zwischen den verschiedenen Zeichensätzen umgeschaltet werden, die wiederum ebenfalls mit Leichtigkeit modifiziert werden können.

Wie kann nun auf die Schriftenvielfalt zugegriffen werden?

1. Mit AvailFonts muß ermittelt werden, welche Zeichensätze zur Verfügung stehen.
2. Mit OpenFont, bzw. OpenDiskFont müssen diese bereitgestellt werden.
3. Mit AddFont werden sie dem System verfügbar gemacht.
4. Mit SetFont wird ein Zeichensatz aktiviert.
5. Mit AskSoftStyle und SetSoftStyle kann seine Darstellungsweise verändert werden.
6. Mit Text können die Zeichen auf den Bildschirm gebracht werden.
7. Mit RemFont können die Zeichensätze dem System "entzogen" werden.
8. Mit CloseFont können Zeichensätze aus dem Speicher entfernt werden.

Der Befehl Text ist der eigentliche Druckbefehl, denn dieser bringt die Zeichen auf den Schirm. Allerdings geht er von der Position aus, an der der Textcursor steht. Dieser Textcursor wird mit den Grafikbefehlen Move, Draw usw. verschoben. Das bedeutet, der Textcursor ist mit dem Grafikstift identisch.

Einzelheiten erfahren Sie aus der folgenden Befehlsbeschreibung und aus dem Demonstrationsprogramm.

### 4.2.1 AddFont

Syntax:           AddFont(textfont)

Funktion:       Hängt einen Zeichensatz an die Zeichensatzliste des Systems an. Dieser Zeichensatz kann anschließend angesprochen werden, wie jeder andere Zeichensatz auch. Das heißt, in einer NewWindow- oder NewScreen-Structure kann dieser Zeichensatz angegeben werden.

Parameter:     textfont           -> Zeiger auf die TextFont-Structure des Zeichensatzes, der angehängt werden soll.

- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct TextFont \*textfont;
- Sonstiges:** Den Zeiger auf die TextFont-Structure bekommt man durch die Befehle OpenFont und OpenDiskFont. Diese Befehle laden Zeichensätze in das System, aber erst durch AddFont sind diese für das System "erreichbar".
- Soll für einen Screen ein eingeladener Zeichensatz verwendet werden, so muß in der NewScreen-Structure zu Beginn ein Zeiger auf die entsprechende TextAttr-Structure gesetzt werden. Diese bekommt man beispielsweise durch den AvailFonts Befehl. Die TextAttr-Structure wird auch für OpenFont und OpenDiskFont benötigt, durch die man einen Zeiger auf die TextFont-Structure erhält.
- Das Ganze mag an dieser Stelle etwas komplex klingen, doch sieht man sich das Demonstrationsprogramm an, wird deutlich, daß das Ganze erheblich einfacher ist, als es hier darzustellen ist.
- Referenz:** Siehe auch OpenDiskFont und OpenFont

## 4.2.2 AskFont

- Syntax:** AskFont(rastport,textattr);
- Funktion:** Dieser Befehl bewirkt das Setzen der TextAttr-Structure mit den Werten des momentan angewählten Zeichensatzes.
- Parameter:** rastport -> Zeiger auf die RastPort-Structure des RastPorts, dessen Zeichensatz abgefragt werden soll.
- textattr -> Zeiger auf eine leere TextAttr-Structure.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct RastPort \*rastport;  
struct TextAttr \*textattr;
- Sonstiges:** Dieser Befehl wird verwendet, um abzufragen, welcher Zeichensatz derzeit in dem spezifizierten RastPort verwendet wird.

Die TextAttr-Structure ist wie folgt aufgebaut:

```
struct TextAttr {
 STRPTR ta_Name; Zeiger auf den Zeichensatzname
 UWORD ta_YSize; Höhe der Zeichen in Pixel
 UBYTE ta_Style; Zeichensatz-Stil (s. AskSoftStyle)
 UBYTE ta_Flags; Zeichensatz-Einstellungen
};
```

Der Zeichensatzname ist bei dem voreingestellten Zeichensatz entweder Topaz-60 oder Topaz-80.

### 4.2.3 AskSoftStyle

Syntax:       maske = AskSoftStyle(rastport);

Funktion:      Ermittelt die momentane Einstellung des Zeichensatzes.

Parameter:    rastport       -> Zeiger auf die RastPort-Structure des RastPorts, für dessen Zeichensatz die Einstellung ermittelt werden soll.

Ergebnis:    maske         -> enthält die Einstellungen des Zeichensatzes.

Datentyp:     struct RastPort \*rastport;  
              BYTE maske;

Sonstiges:    Jeder Zeichensatz kann mit SetSoftStyle noch verändert werden. Wie der Zeichensatz momentan eingestellt ist, kann mit AskSoftStyle ermittelt werden.

»maske« ist vom Typ BYTE, wobei nur die unteren 4 Bits verwendet werden:

```
00000000 = 0 <-> Normal
00000001 = 1 <-> Unterstrichen
00000010 = 2 <-> Fettdruck
00000100 = 4 <-> Schrägschrift
00001000 = 8 <-> Breitschrift
```

Das bedeutet, wenn »maske« gleich 5 ist, werden die Zeichen unterstrichen und schräg gedruckt.

### 4.2.4 AvailFonts

Syntax:       fehler = AvailFonts(puffer,bytes,typen);

Funktion:      Ermittelt alle erreichbaren Zeichensätze aus dem Speicher und von der Diskette.

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                  |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameter: | puffer                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | -> Zeiger auf den Speicherbereich, in den eine AvailFontsHeader-Structure und für jeden Zeichensatz jeweils eine AvailFont-Structure eingetragen wird.                                                                                                                                                                                           |
|            | bytes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | -> Größe des Puffers in Bytes.                                                                                                                                                                                                                                                                                                                   |
|            | typen                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | -> Hier muß AFF_MEMORY angegeben werden, wenn nur im Speicher nach den Zeichensätzen gesucht werden soll.<br><br>AFF_DISK kann angegeben werden, wenn nur auf Diskette nach den Zeichensätzen gesucht werden soll.<br><br>Wird AFF_MEMORY AFF_DISK angegeben, wird sowohl im Speicher, als auch auf der Diskette nach den Zeichensätzen gesucht. |
| Ergebnis:  | fehler                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | -> ist 0, wenn alles in Ordnung war. Ist fehler ungleich 0, gibt diese Variable die Anzahl der Bytes an, um die der Puffer vergrößert werden muß.                                                                                                                                                                                                |
| Datentyp:  | ULONG puffer;<br>int bytes, typen;<br>int fehler;                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                  |
| Sonstiges: | Die AvailFontsHeader-Structure ist äußerst einfach aufgebaut. Sie besteht nur aus einem einzigen Eintrag, der Anzahl der gefundenen Zeichensätze. Dieser Eintrag ist vom Typ WORD, kann also einfach abgefragt werden, in dem die ersten zwei Bytes des Puffers ermittelt werden. Diesem schließt sich eine Anzahl von AvailFonts-Structures an, die folgendermaßen aufgebaut sind:<br><br><pre>struct AvailFonts {     UWORD af_Type;           AFF_MEMORY oder AFF_DISK     struct TextAttr af_Attr; TextAttr-Structure };</pre><br>Für jeden gefundenen Zeichensatz ist eine AvailFonts-Structure vorhanden. Ist "af_Type" gleich AFF_MEMORY, ist der Zeichensatz schon im Speicher, bei AFF_DISK, muß er noch von Diskette geladen werden, um ihn zu benutzen. Die TextAttr-Structure wird für OpenFont und OpenDiskFont benötigt. |                                                                                                                                                                                                                                                                                                                                                  |

## 4.2.5 ClearEOL

|            |                                                                                                       |
|------------|-------------------------------------------------------------------------------------------------------|
| Syntax:    | ClearEOL(rastport);                                                                                   |
| Funktion:  | Löscht von der derzeitigen Textcursor-Position in dem angegebenen RastPort bis zum Ende der Zeile.    |
| Parameter: | rastport -> Zeiger auf die RastPort-Structure des RastPorts, in dem gelöscht werden soll.             |
| Ergebnis:  | Kein Ergebnis.                                                                                        |
| Datentyp:  | struct RastPort *rastport;                                                                            |
| Sonstiges: | Der freiwerdende Bereich wird mit der Hintergrundfarbe, die mit SetBPen gesetzt werden kann, gefüllt. |
| Referenz:  | Siehe auch ClearScreen                                                                                |

## 4.2.6 ClearScreen

|            |                                                                                                      |
|------------|------------------------------------------------------------------------------------------------------|
| Syntax:    | ClearScreen(rastport);                                                                               |
| Funktion:  | Löscht den gesamten Bereich, der durch das RastPort definiert ist.                                   |
| Parameter: | rastport -> Zeiger auf die RastPort-Structure des RastPorts, der gelöscht werden soll.               |
| Ergebnis:  | Kein Ergebnis.                                                                                       |
| Datentyp:  | struct RastPort *rastport;                                                                           |
| Sonstiges: | Der freiwerdende Bereich wird in der Hintergrundfarbe, die mit SetBPen gesetzt werden kann, gefüllt. |
| Referenz:  | Siehe auch ClearEOL                                                                                  |

## 4.2.7 CloseFont

|            |                                                                                                                                                                                     |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | CloseFont(textfont);                                                                                                                                                                |
| Funktion:  | Schließt einen, mit OpenFont oder OpenDiskFont geöffneten, Zeichensatz und löscht ihn aus dem Speicher, was besonders dann zu empfehlen ist, wenn der Speicherplatz zur Neige geht. |
| Parameter: | textfont -> Zeiger auf die TextFont-Structure des Zeichensatzes, der nicht mehr benötigt wird.                                                                                      |

|            |                                                                                                                                                          |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ergebnis:  | Kein Ergebnis.                                                                                                                                           |
| Datentyp:  | struct TextFont *textfont;                                                                                                                               |
| Sonstiges: | Der Zeichensatz wird natürlich nicht von der Diskette gelöscht!<br><br>Zu der TextFont-Structure gelangt man über die OpenFont und OpenDiskFont Befehle. |
| Referenz:  | Siehe auch OpenFont und OpenDiskFont                                                                                                                     |

### 4.2.8 **OpenDiskFont**

|            |                                                                                                                          |
|------------|--------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | textfont = OpenDiskFont(textattr);                                                                                       |
| Funktion:  | Holt einen, mit textattr spezifizierten, Zeichensatz von der Diskette in den Speicher.                                   |
| Parameter: | textattr           -> Zeiger auf die TextAttr-Structure des Zeichensatzes, der geladen werden soll.                      |
| Ergebnis:  | textfont           -> Zeiger auf die TextFont-Structure des Zeichensatzes, die beispielsweise für AddFont benötigt wird. |
| Datentyp:  | struct TextAttr *textattr;<br>struct TextFont *textfont;                                                                 |
| Sonstiges: | Dieser Befehl lädt nur einen Zeichensatz in den Speicher. Verfügbar ist er erst nach dem AddFont Befehl.                 |
| Referenz:  | Siehe auch OpenFont und AddFont                                                                                          |

## 4.2.9 OpenFont

|            |                                                                                                              |
|------------|--------------------------------------------------------------------------------------------------------------|
| Syntax:    | <code>textfont = OpenFont(textattr);</code>                                                                  |
| Funktion:  | Sucht nach einem, im Speicher abgelegten, Zeichensatz.                                                       |
| Parameter: | <code>textattr</code> -> Zeiger auf die TextAttr-Structure des zu suchenden Zeichensatzes.                   |
| Ergebnis:  | <code>textfont</code> -> Zeiger auf die TextFont-Structure des Zeichensatzes, die für AddFont benötigt wird. |
| Datentyp:  | <code>struct TextAttr *textattr;</code><br><code>struct TextFont *textfont;</code>                           |
| Sonstiges: | Erst wenn anschließend der AddFont Befehl gegeben wird, kann der spezifizierte Zeichensatz verwendet werden. |
| Referenz:  | Siehe auch OpenDiskFont und AddFont                                                                          |

## 4.2.10 RemFont

|            |                                                                                                                                              |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | <code>fehler = RemFont(textfont);</code>                                                                                                     |
| Funktion:  | Löscht den spezifizierten Zeichensatz aus der Zeichensatzliste des Systems, wodurch er für dieses nicht mehr verfügbar ist.                  |
| Parameter: | <code>textfont</code> -> Zeiger auf die TextFont-Structure des zu entfernenden Zeichensatzes.                                                |
| Ergebnis:  | <code>fehler</code> -> ist 0, wenn kein Fehler auftrat.                                                                                      |
| Datentyp:  | <code>struct TextFont *textfont;</code><br><code>int fehler;</code>                                                                          |
| Sonstiges: | Dieser Befehl löscht den Zeichensatz nicht aus dem Speicher. Soll dies geschehen, muß anschließend noch der Befehl CloseFont gegeben werden. |

## 4.2.11 SetFont

|           |                                                                     |
|-----------|---------------------------------------------------------------------|
| Syntax:   | <code>fehler = SetFont(rastport,textfont);</code>                   |
| Funktion: | Ordnet dem spezifizierten RastPort einen bestimmten Zeichensatz zu. |

|            |                                                                                                           |                                                                                                  |
|------------|-----------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| Parameter: | rastport                                                                                                  | -> Zeiger auf die RastPort-Structure des RastPorts, dem ein Zeichensatz zugeordnet werden soll.  |
|            | textfont                                                                                                  | -> Zeiger auf die TextFont-Structure des Zeichensatzes, der dem RastPort zugeordnet werden soll. |
| Ergebnis:  | fehler                                                                                                    | -> ist 0, wenn kein Fehler aufgetreten ist.                                                      |
| Datentyp:  | struct RastPort *rastport;<br>struct TextFont *textfont;<br>int fehler;                                   |                                                                                                  |
| Sonstiges: | Nach diesem Befehl werden sämtliche Textausgaben in diesem RastPort mit dem neuen Zeichensatz ausgegeben. |                                                                                                  |

## 4.2.12 SetSoftStyle

|            |                                                          |                                                                                                                                             |
|------------|----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | neumaske = SetSoftStyle(rastport,neu,maske);             |                                                                                                                                             |
| Funktion:  | Setzt einen neuen Darstellungsmodus.                     |                                                                                                                                             |
| Parameter: | rastport                                                 | -> Zeiger auf die RastPort-Structure des RastPorts, für den ein neuer Darstellungsmodus gesetzt werden soll.                                |
|            | neu                                                      | -> Eine Byte-Variable, die den neuen Darstellungsmodus enthält. Wie sie gesetzt werden kann, ist unter dem Befehl AskSoftStyle beschrieben. |
|            | maske                                                    | -> gibt an, welche Bits verändert werden dürfen. Diese Variable wird von einem Aufruf AskFontStyle zurückgegeben (siehe dort).              |
| Ergebnis:  | neumaske                                                 | -> hat die gleiche Funktion wie »maske«, jedoch sind die Bit's, die durch »neu« bestimmt wurden, schon gesetzt.                             |
| Datentyp:  | struct RastPort *rastport;<br>BYTE neu, maske, neumaske; |                                                                                                                                             |
| Sonstiges: | Für weitere Einzelheiten siehe AskSoftStyle.             |                                                                                                                                             |
| Referenz:  | Siehe auch AskSoftStyle                                  |                                                                                                                                             |

### 4.2.13 Text

|            |                                                                                                                                        |                                                                                     |
|------------|----------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| Syntax:    | <code>fehler = Text(rastport,&amp;strptr[0],länge);</code>                                                                             |                                                                                     |
| Funktion:  | Schreibt den angegebenen Text in das RastPort, an die momentane Position des Textcursors.                                              |                                                                                     |
| Parameter: | <code>rastport</code>                                                                                                                  | -> Zeiger auf die RastPort-Structure des RastPorts, in den geschrieben werden soll. |
|            | <code>&amp;strptr[0]</code>                                                                                                            | -> Zeiger auf den Text, der ausgegeben werden soll.                                 |
|            | <code>länge</code>                                                                                                                     | -> Anzahl der auszugebenden Buchstaben.                                             |
| Ergebnis:  | <code>fehler</code>                                                                                                                    | -> ist 0, wenn kein Fehler auftrat.                                                 |
| Datentyp:  | <code>struct RastPort *rastport;</code><br><code>char strptr[];</code><br><code>int länge;</code>                                      |                                                                                     |
| Sonstiges: | Mit diesem Befehl muß vorsichtig umgegangen werden, da unerwünschte Effekte auftreten können, falls der Text über den Rand hinausragt. |                                                                                     |

### 4.2.14 TextLength

|            |                                                               |                                                                                                  |
|------------|---------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| Syntax:    | <code>pix = TextLength(rastport,&amp;strptr[0],länge);</code> |                                                                                                  |
| Funktion:  | Ermittelt die Länge in Pixel für den angegebenen Text.        |                                                                                                  |
| Parameter: | <code>rastport</code>                                         | -> Zeiger auf die RastPort-Structure des RastPorts, für die die Textlänge ermittelt werden soll. |
|            | <code>&amp;strptr[0]</code>                                   | -> Zeiger auf den Text, dessen Länge ermittelt werden soll.                                      |
|            | <code>länge</code>                                            | -> Anzahl der Zeichen, die dieser Text enthält.                                                  |
| Ergebnis:  | <code>pix</code>                                              | -> Anzahl der Grafikpunkte, die dieser Text benötigt.                                            |

**Datentyp:**        `struct RastPort *rastport;`  
                   `char strptr[];`  
                   `int länge;`  
                   `int pix;`

**Sonstiges:**        Dieser Befehl ermittelt nicht die Anzahl der Buchstaben, die der Text enthält, denn diese müssen beim Aufruf schon mit-angegeben werden.

```

1 /*****
2
3 Text-Demonstration
4 last update 26/05/87
5 von Frank Kremser und Joerg Koch
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Darstellen 5 verschiedener Fonts, die sich auf der Diskette befinden,
11 mit Hilferder Intuitext-Funktion.
12
13 *****/
14
15 #include "exec/types.h" /* Include-Files werden eingelesen */
16 #include "exec/io.h"
17 #include "exec/memory.h"
18 #include "exec/exec.h"
19 #include "graphics/gfx.h"
20 #include "hardware/dmabits.h"
21 #include "hardware/custom.h"
22 #include "hardware/blit.h"
23 #include "graphics/gfxmacros.h"
24 #include "graphics/copper.h"
25 #include "graphics/view.h"
26 #include "graphics/gels.h"
27 #include "graphics/regions.h"
28 #include "graphics/clip.h"
29 #include "graphics/text.h"
30 #include "graphics/gfxbase.h"
31 #include "devices/keymap.h"
32 #include "libraries/dos.h"
33 #include "graphics/text.h"
34 #include "intuition/intuition.h"
35 #include "libraries/diskfont.h"
36
37 struct TextFont *textfont; /* Font-Zeiger */
38 struct TextAttr textattr;
39
40 struct IntuitionBase *IntuitionBase; /* Lib-Zeiger */
41 struct GfxBase *GfxBase;
42 ULONG DiskfontBase;
43
44 struct RastPort *rp;
45 struct NewScreen ns = /* New-Screen Struktur */
46 (
47 0, /* linke Ecke */
48 0, /* obere Ecke */
49 640, /* Breite */
50 256, /* Hoehe */

```

```

51 2, /* Tiefe */
52 0, /* DetailPen */
53 1, /* BlockPen */
54 HIRES, /* ViewModes */
55 CUSTOMSCREEN, /* Type */
56 NULL,
57 "Text-Demonstration", /* Screen Titel */
58 NULL,
59 NULL
60 };
61
62
63 main()
64 {
65 struct Screen *screen;
66 LONG warte; /* oeffnen der Libraries */
67
68 IntuitionBase = (struct IntuitionBase *)
69 OpenLibrary("intuition.library",0);
70 if(IntuitionBase == NULL) exit();
71
72 GfxBase = (struct GfxBase *)
73 OpenLibrary("graphics.library",0);
74 if(GfxBase == NULL) exit();
75
76 DiskfontBase = OpenLibrary("diskfont.library",0);
77 if(DiskfontBase == NULL) exit();
78
79 if ((screen = (struct Screen *) /* oeffnen des Screens */
80 OpenScreen(&ns)) == NULL) exit();
81
82 rp = &screen->RastPort;
83
84 textattr.ta_Name = "ruby.font"; /* Struktur zur Festlegung */
85 textattr.ta_YSize = 12; /* der Font-Art deklarieren */
86 textattr.ta_Style = 8; /* Schriftart */
87 textattr.ta_Flags = 0;
88
89 textfont = (struct TextFont *)OpenDiskFont(&textattr);
90 if(textfont != 0) /* Diskfont oeffnen */
91 {
92 SetFont(rp,textfont); /* Schrift setzen */
93 Move(rp,0,40); /* Font zeichnen und schliessen */
94 Text(rp,"Dies ist Ruby 12",16); /* Text ausgeben */
95 CloseFont(textfont); /* Font schliessen */
96 };
97
98 textattr.ta_Name = "emerald.font"; /* naechster Font */
99 textattr.ta_YSize = 20;
100 textattr.ta_Style = 1;
101 textattr.ta_Flags = 0;
102 textfont = (struct TextFont *)OpenDiskFont(&textattr);
103 if(textfont != 0)
104 {
105 SetFont(rp,textfont);
106 Move(rp,0,60);
107 Text(rp,"Dies ist Emerald 20",19);
108 CloseFont(textfont);
109 };
110
111 textattr.ta_Name = "opal.font"; /* noch ein Font */
112 textattr.ta_YSize = 12;
113 textattr.ta_Style = 4;

```

```
114 textattr.ta_Flags = 0;
115 textfont = (struct TextFont *)OpenDiskFont(&textattr);
116 if(textfont != 0)
117 {
118 SetFont(rp,textfont);
119 Move(rp,0,80);
120 Text(rp,"Dies ist Opal 12",16);
121 CloseFont(textfont);
122 };
123
124 textattr.ta_Name = "garnet.font"; /* naechster Font */
125 textattr.ta_YSize = 16;
126 textattr.ta_Style = 8;
127 textattr.ta_Flags = 0;
128 textfont = (struct TextFont *)OpenDiskFont(&textattr);
129 if(textfont != 0)
130 {
131 SetFont(rp,textfont);
132 Move(rp,0,100);
133 Text(rp,"Dies ist Garnet 16",18);
134 CloseFont(textfont);
135 };
136
137 textattr.ta_Name = "sapphire.font"; /* letzter Font */
138 textattr.ta_YSize = 19;
139 textattr.ta_Style = 3;
140 textattr.ta_Flags = 0;
141 textfont = (struct TextFont *)OpenDiskFont(&textattr);
142 if(textfont != 0)
143 {
144 SetFont(rp,textfont);
145 Move(rp,0,120);
146 Text(rp,"Dies ist Sapphire 19",20);
147 CloseFont(textfont);
148 };
149
150 for(warte = 0; warte < 1000000; warte++);
151
152 CloseScreen(screen); /* Schleife bis 1000000, dann */
153 CloseLibrary(IntuitionBase); /* Libs schliessen */
154 CloseLibrary(GfxBase);
155 CloseLibrary(DiskfontBase);
156 }
```

## 4.3 Die Images

Für viele Anwendungen werden Images benötigt. So kann einem Gadget etwa ein eigenes Image zugewiesen werden. Dieses Image besteht aus Informationen über die Form, die Farbe und die Höhe.

Um ein Image benutzen zu können, muß allerdings zuerst eine Image-Structure erstellt werden. Diese hat folgende Form:

```
struct Image
{
 SHORT LeftEdge, TopEdge; Koord. der linken, oberen Ecke
 SHORT Width, Height, Depth; Breite, Höhe und Tiefe des Images
 SHORT *ImageData; Zeiger auf die Imagedaten
 UBYTE PlanePick, PlaneOnOff; Angaben für die Farben
 struct Image *NextImage; Zeiger auf ein weiteres Image
};
```

Die Angaben über die linke, obere Ecke werden später zu den Angaben, die in der Gadget-Structure oder bei DrawImage stehen, hinzuaddiert. Die Höhe und Breite des Image muß in Pixel angegeben werden. Die Tiefe gibt an, wieviele BitPlanes das Image belegt. Der Zeiger auf das nächste Image kann im Normalfall auf »NULL« gesetzt werden.

Im Gegensatz zu den Sprites müssen bei Images bei der Definition der Image-Daten die Daten für die erste Plane und die zweite Plane getrennt werden.

Wir wollen an dieser Stelle ein Rechteck definieren, das 16 Pixel breit und 4 Pixel hoch ist:



■ bedeutet, daß BitPlane 0 verwendet wird.  
 ■■■ bedeutet, daß BitPlane 1 verwendet wird.

Daraus ergeben sich für BitPlane 0 folgende Werte:

```
0xFFFF
0x8000
0x8000
0x8000
```

Für BitPlane 1 ergeben sich dann die Werte

```
0x0000
0x0001
0x0001
0xFFFF
```

Die Image-Daten müssen also folgendermaßen deklariert werden:

```
USHORT data[] =
{
 0xFFFF, /* erste Plane */
 0x8000,
 0x8000,
 0x8000,

 0x0000, /* zweite Plane */
 0x0001,
 0x0001,
 0xFFFF
};
```

Die Image-Structure hat im Programm dann folgendes Aussehen:

```
struct Image ownimage =
{
 0,
 0,
 16,
 4,
 2,
 &data[0],
 3,
 0,
 NULL
};
```

Mit dem Befehl `DrawImage(RastPort,&ownimage,x,y);` kann das Image auch direkt auf den Bildschirm gezeichnet werden.

`RastPort` ist ein Zeiger auf die `RastPort`-Structure des `RastPorts`, in den gezeichnet werden soll. `x` und `y` geben die Position an, an der das Image gezeichnet werden soll. `x` und `y` werden zu `LeftEdge` und `TopEdge` hinzuaddiert.

`PlanePick` bezeichnet im übrigen, welche `BitPlanes` gesetzt werden sollen, wenn ein Punkt im Image gesetzt ist. `PlaneOnOff` hat den gleichen Zweck für die Punkte, die nicht gesetzt sind.

Folgende Werte können `PlanePick` und `PlaneOnOff` bei einem Screen mit drei `BitPlanes` annehmen – Für einen Screen mit mehr `BitPlanes` können Sie die Folge immer weiter führen, die Regel erkennen Sie sicher leicht aus der folgenden Reihe –:

- 0 -> Keine Planes sollen gesetzt werden.
- 1 -> Plane 0 soll gesetzt werden.
- 2 -> Plane 1 soll gesetzt werden.
- 3 -> Planes 0 und 1 sollen gesetzt werden.
- 4 -> Plane 2 soll gesetzt werden.
- 5 -> Planes 0 und 2 sollen gesetzt werden.
- 6 -> Planes 1 und 2 sollen gesetzt werden.
- 7 -> Planes 0, 1 und 2 sollen gesetzt werden.

Da unser Image zwei Bitplanes verwendet, kommen für PlanePick also nur die Werte 3, 5 und 6 in Betracht. Für PlaneOnOff wählen wir 0, da keine Farbe gesetzt werden soll.

## 4.4 Umrahmungen: Borders

Borders werden für Umrahmungen eingesetzt. Für die Verwendung muß eine Anzahl von Eckpunkten angegeben werden, die durch Linien verbunden werden. Um sie einsetzen zu können, muß als erstes eine Border-Structure deklariert werden. Diese hat folgende Form:

```
struct Border
{
 SHORT LeftEdge, TopEdge; Koord. der linken, oberen Ecke
 SHORT FrontPen, BackPen, DrawMode; Farben und Zeichenmodus
 SHORT Count; Anzahl der Ecken
 SHORT *XY; Zeiger auf die Eckdaten
 struct Border * NextBorder; Zeiger auf nächste Border
}
```

Die Angaben über die linke, obere Ecke werden später zu den Angaben, die in der Gadget-Structure oder bei DrawBorder stehen, hinzuaddiert. FrontPen gibt die Farbe an, in der die Border normalerweise gezeichnet wird. Wird für DrawMode nicht JAM1, sondern JAM2 gewählt, so wird auch die Farbe, die mit BackPen angegeben ist, verwendet (nähere Angaben unter SetDrMd). Der Zeiger auf die nächste Border kann im Normalfall auf »NULL« gesetzt werden.

Count gibt an, wieviele Eckpunkte die Border besitzt. XY ist der Zeiger auf das Feld, in dem die Koordinaten der Eckpunkte eingetragen sind.

Wir wollen nun eine Border erstellen. Als erstes müssen die Eckpunkte angegeben werden:

```
USHORT data[] =
{
 100,100, /* Startpunkt */
 100,200,
 200,200,
 200,100,
 100,100 /* und zurück zum Startpunkt */
};
```

Anschließend wird die Border-Structure erstellt:

```
struct Border rahmen =
{
 0, /* Linke und Obere Ecke würden zu den Border-Daten */
 0, /* hinzuaddiert werden */
 3,
 0,
 JAM1,
 5,
 &data[0],
 NULL
};
```

Mit `DrawBorder(RastPort,&rahmen,x,y)`; kann die Border dann direkt auf den Bildschirm gezeichnet werden.

`RastPort` ist ein Zeiger auf die `RastPort`-Structure des `RastPorts`, in den gezeichnet werden soll. `x` und `y` geben die Position an, an der die Border gezeichnet werden soll. `x` und `y` werden zu `LeftEdge` und `TopEdge` und den Punktkoordinaten hinzuaddiert.

```

1 /*****
2
3 Image- und Border-Demonstration
4 last update 26/05/87
5 von Joerg Koch und Frank Kremser
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Diese Demonstration zeigt die Anwendungsmoeglichkeiten von Images und
11 Borders. Auch Images koennen ohne weiteres zur Animation verwendet werden.
12
13 *****/
14
15
16 #include "exec/types.h" /* Einladen der Include-Files */
17 #include "exec/io.h"
18 #include "exec/memory.h"
19 #include "exec/exec.h"
20 #include "graphics/gfx.h"
21 #include "hardware/dmabits.h"
22 #include "hardware/custom.h"
23 #include "hardware/blit.h"
24 #include "graphics/gfxmacros.h"
25 #include "graphics/copper.h"
26 #include "graphics/view.h"
27 #include "graphics/gels.h"
28 #include "graphics/regions.h"
29 #include "graphics/clip.h"
30 #include "graphics/text.h"
31 #include "graphics/gfxbase.h"
32 #include "devices/keymap.h"
33 #include "libraries/dos.h"
34 #include "graphics/text.h"
35 #include "intuition/intuition.h"
36 #include "libraries/diskfont.h"
37
38 struct IntuitionBase *IntuitionBase; /* Lib - Zeiger */
39 struct GfxBase *GfxBase;
40
41 USHORT data1[] = /* Die verschiedenen Images */
42 {
43 0x03C0, 0x0FF0,
44 0x1FFB, 0x3FFC,
45 0x7FFE, 0x7FFE,
46 0xFFFF, 0xFFFF,
47 0xFFFF, 0xFFFF,
48 0x7FFE, 0x7FFE,
49 0x3FFC, 0x1FFB,
50 0x0FF0, 0x03C0,
51
52 0x0000, 0x0200,

```

```
53 0x0200, 0x0400,
54 0x0400, 0x0800,
55 0x0800, 0x1000,
56 0x1000, 0x0800,
57 0x0800, 0x0400,
58 0x0400, 0x0200,
59 0x0200, 0x0000
60 };
61
62 struct Image image1 =
63 {
64 0,
65 0,
66 16,
67 16,
68 2,
69 &data1[0],
70 3,
71 0,
72 NULL.
73 };
74
75 USHORT data2[] =
76 {
77 0x03C0, 0x0FF0,
78 0x1FFB, 0x3FFC,
79 0x7FFE, 0x7FFE,
80 0xFFFF, 0xFFFF,
81 0xFFFF, 0xFFFF,
82 0x7FFE, 0x7FFE,
83 0x3FFC, 0x1FFB,
84 0x0FF0, 0x03C0,
85
86 0x0000, 0x0100,
87 0x0100, 0x0100,
88 0x0100, 0x0100,
89 0x0100, 0x0100,
90 0x00B0, 0x00B0,
91 0x00B0, 0x00B0,
92 0x00B0, 0x00B0,
93 0x00B0, 0x00B0
94 };
95
96 struct Image image2 =
97 {
98 0,
99 0,
100 16,
101 16,
102 2,
103 &data2[0],
104 3,
105 0,
106 NULL
107 };
108
109 USHORT data3[] =
110 {
111 0x03C0, 0x0FF0,
112 0x1FFB, 0x3FFC,
113 0x7FFE, 0x7FFE,
114 0xFFFF, 0xFFFF,
115 0xFFFF, 0xFFFF,
116 0x7FFE, 0x7FFE,
```

```
117 0x3FFC, 0x1FFB,
118 0x0FF0, 0x03C0,
119
120 0x0000, 0x0040,
121 0x0040, 0x0020,
122 0x0020, 0x0010,
123 0x0010, 0x0008,
124 0x0008, 0x0010,
125 0x0010, 0x0020,
126 0x0020, 0x0040,
127 0x0040, 0x0000
128 };
129
130 struct Image image3 =
131 {
132 0,
133 0,
134 16,
135 16,
136 2,
137 &data3[0],
138 3,
139 0,
140 NULL
141 };
142
143 USHORT bordata[] = /* Die Border um das Feld */
144 {
145 0, 0,
146 0, 200,
147 190, 200,
148 190, 0,
149 0, 0
150 };
151
152 struct Border border =
153 {
154 0,
155 0,
156 3,
157 2,
158 JAM1,
159 5,
160 &bordata[0],
161 NULL
162 };
163
164 struct RastPort *rp;
165
166 struct NewScreen ns = /* Der eigene Screen */
167 {
168 0,
169 0,
170 320,
171 256,
172 2,
173 0,
174 1,
175 0,
176 CUSTOMSCREEN,
177 NULL,
178 "Image-/Border-Demonstration",
179 NULL,
```

```
180 NULL
181 };
182
183
184 main()
185 {
186 struct Screen *screen;
187 LONG schleife;
188 int x,y;
189
190 IntuitionBase = (struct IntuitionBase *)
191 OpenLibrary("intuition.library",0);
192 if(IntuitionBase == NULL) exit(); /* oeffnen der Libs */
193
194 GfxBase = (struct GfxBase *)
195 OpenLibrary("graphics.library",0);
196 if(GfxBase == NULL) exit();
197
198 if ((screen = (struct Screen *)
199 OpenScreen(&ns)) == NULL) exit(); /* oeffnen des Screens */
200
201 rp = &screen->RastPort;
202
203 for(schleife = 0; schleife < 6; schleife++) /* Border zeichnen */
204 DrawBorder(rp,&border,70 + schleife,15 + schleife);
205
206 for(schleife = 0; schleife < 50; schleife++) /* Animation mit Images */
207 { /* durch verschiedene */
208 for(x = 0; x < 9; x++) /* Images */
209 for(y = 0; y < 9; y++)
210 DrawImage(rp,&image1,x * 20 + 80,y * 20 + 28);
211 for(x = 0; x < 9; x++)
212 for(y = 0; y < 9; y++)
213 DrawImage(rp,&image2,x * 20 + 80,y * 20 + 28);
214 for(x = 0; x < 9; x++)
215 for(y = 0; y < 9; y++)
216 DrawImage(rp,&image3,x * 20 + 80,y * 20 + 28);
217 };
218
219 for(schleife = 0; schleife < 200000; schleife++);
220
221 CloseScreen(screen); /* schliessen des Screens und der Libs */
222 CloseLibrary(IntuitionBase);
223 CloseLibrary(GfxBase);
224 }
```

## 4.5 Intuition-Text

Intuition stellt auch eine Funktion zur Verfügung, um Text auf den Bildschirm zu bringen. Aber auch für eine Vielzahl anderer Anwendungen wird Intuition-Text benötigt, zum Beispiel für Menu's.

Als erstes muß eine IntuiText-Structure deklariert werden. Diese hat folgende Form:

```
struct IntuiText
{
 UBYTE FrontPen, BackPen; Farben für den Text
 UBYTE DrawMode; Schreib-Modus
 SHORT LeftEdge, TopEdge; Textposition
 struct TextAttr *ITextFont; Zeichensatz
 UBYTE *IText; Auszugebender Text
 struct IntuiText *NextText; Zeiger auf nächsten Text
};
```

Die Angaben über die linke, obere Ecke geben die Position des ersten Zeichens an, das ausgegeben werden soll. FrontPen gibt die Farbe an, in der der Text normalerweise geschrieben wird. Wird für DrawMode nicht JAM1, sondern JAM2 gewählt, so wird auch die Farbe, die mit BackPen angegeben ist, verwendet (nähere Angaben unter SetDrMd). Der Zeiger auf den Zeichensatz, der verwendet werden soll, kann auf »NULL« gesetzt werden, wenn der eingestellte Zeichensatz verwendet werden soll. Der Zeiger auf den nächsten Text kann im Normalfall auf »NULL« gesetzt werden.

Wie man zu der TextAttr-Structure für einen anderen Zeichensatz gelangt, ist unter den Text-Funktionen nachzulesen.

Wir wollen nun also einen Text ausgeben. Dazu deklarieren wir eine IntuiText-Structure:

```
struct IntuiText text =
{
 2,
 0,
 JAM1,
 100,
 50,
 NULL,
 "Text-Demonstration",
 NULL
};
```

Mit PrintIText(RastPort,&text,x,y); kann der Text nun einfach auf dem Bildschirm ausgegeben werden.

RastPort ist ein Zeiger auf die RastPort-Structure des RastPorts, in den geschrieben werden soll. x und y geben die Position an, an der der Text geschrieben werden soll. x und y werden zu LeftEdge und TopEdge hinzugeaddiert.

# Einfache Animationen in Screens und Windows

Das Amiga-System stellt eine Vielzahl von Animations-Möglichkeiten zur Verfügung. Diese reichen von den bekannten Sprites über AnimObjects bis hin zu den weniger bekannten Möglichkeiten durch Preferences und Playfields. Wie diese Möglichkeiten angewendet werden, möchten wir in diesem Kapitel erläutern.

Folgende Animationen stehen zur Verfügung:

- Sprites: Einfache Sprites, wie sie auch von anderen Computern bekannt sind. Das Amiga-System unterstützt maximal 8 Hardware-Sprites, die 16 Punkte breit und beliebig hoch sein dürfen. Dabei dürfen sie maximal 4 Farben enthalten, wobei eine »Farbe« durchsichtig ist.
- VSprites: VSprites, oder auch virtuelle Sprites, sind ähnlich zu handhaben wie Hardware-Sprites, erlauben aber noch zusätzliche Abfragen, wie zum Beispiel die Kollisionsabfrage. VSprites sind im Prinzip auch Hardware-Sprites, da sie für die Darstellung auf diese zurückgreifen.
- Bobs: Bobs, oder auch Blitter-Objekte, sind Software-Sprites, die direkt in den Screen gezeichnet werden. Dies vollzieht sich durch den Blitter zwar enorm schnell, aber sie sind doch langsamer als Hardware-Sprites, was besonders bei größeren Bobs deutlich wird. Sie unterliegen nur den Beschränkungen des Screens.
- AnimObjects: AnimObjects bestehen aus einer Vielzahl von VSprites und/oder Bobs, die in Folge gezeigt werden und so einen Bewegungsablauf suggerieren können.

- Maus: Für einfache Animationen kann auch der Mauszeiger herangezogen werden. Durch SetPointer kann dieser verändert werden.
- Preferences: Über Preferences kann der Bildschirm als Ganzes verschoben werden.
- Playfields: Playfields ermöglichen zudem, Grafiken mit einer Auflösung von bis zu 1024 x 1024 Pixels zu verwenden, aus der aber immer nur ein Teil gezeigt wird. Durch Verschieben der Playfields kann ebenfalls Animation erzielt werden.

## 5.1 Einfache Hardware-Sprites

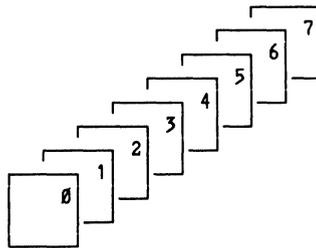
Der Amiga stellt hardwaremäßig maximal 8 Sprites zur Verfügung. Diese Sprites dürfen 16 Punkte breit und beliebig hoch sein. Sie werden allerdings in der niedrigen Auflösung dargestellt, was auch dann gilt, wenn ein hochauflösender Screen verwendet wird. Ein Beispiel dafür ist der Mauszeiger, der nach dem Einschalten auf einem 640 x 256 Pixel-Screen gezeigt wird.

Sprites dürfen bis zu 4 Farben verwenden, die aus den Screenfarbregistern entnommen werden. Folgende Sprite-Farbkombinationen sind vorhanden:

|                   |                        |
|-------------------|------------------------|
| Sprite 0 und 1 -> | Farbregister 16 bis 19 |
| 2 und 3 ->        | 20 bis 23              |
| 4 und 5 ->        | 24 bis 27              |
| 6 und 7 ->        | 28 bis 31              |

Die Farbregister 16, 20, 24 und 28 haben allerdings keine Bedeutung, da die Sprite-Pixel, die auf diese Farbregister zugreifen, durchsichtig sind. Diese Farbregister gelten auch für Screens, die keine 32 Farben verwenden. Aus dieser Zuweisung folgt, daß einige Screenfarben unter Umständen mit den Spritefarben übereinstimmen, was aber bei geschickter Farbverteilung keine Einschränkung bedeuten sollte.

Zudem haben die Sprites eine bestimmte Priorität, die nicht außer Kraft gesetzt werden kann:



Das bedeutet, daß Sprite 2 immer vor Sprite 4 erscheint, wenn sie sich überlappen.

Um ein Sprite auf den Schirm zu bekommen, muß als erstes in der NewScreen-Structure das Flag »SPRITES« gesetzt werden, was für den Workbench-Screen nicht nötig ist. Anschließend muß eine SpriteImage- und eine SimpleSprite-Structure initialisiert werden. Diese Structures haben folgende Form:

```

struct SpriteImage
{
 UWORD posctldata[2]; /* Positionskontrolle */
 UWORD sprdata[2][height]; /* Spritedaten */
 UWORD reserved[2]; /* reserviert für spätere Anwendungen */
};

struct SimpleSprite
{
 UWORD *posctldata; /* Zeiger auf SpriteImage-Structure */
 UWORD height; /* Höhe des Sprites */
 UWORD x,y; /* Position des Sprites */
 UWORD num; /* Sprite-Nummer */
};

```

Wie die SimpleSprite-Structure initialisiert wird, ist aus dem Demonstrationsprogramm ersichtlich.

Auf die SpriteImage-Structure wollen wir hier etwas näher eingehen:

Wir wollen ein Sprite initialisieren, das 5 Punkte hoch und 16 Punkte breit ist. Es soll ein Rechteck darstellen, dessen vier Seiten die vier Farben repräsentieren:

```

1111111111111110 erste Zeile = 0xFFFE
1000000000000001 erste Zeile = 0x8001

1000000000000000 zweite Zeile = 0x8000
1000000000000001 zweite Zeile = 0x8001

1000000000000000 usw. = 0x8000
1000000000000001 = 0x8001

1000000000000000 = 0x8000
1000000000000001 = 0x8001

1000000000000000 = 0x8000
1000000000000001 = 0x8001

```

```

0 bedeutet Farbe 1 1
0 (= durchsichtig) 0 bedeutet Farbe 2

0 1
1 bedeutet Farbe 3 1 bedeutet Farbe 4

```

Daraus ergibt sich folgende SpriteImage-Structure:

```

struct SpriteImage *SprIm =
{
 0,0, /* Positionskontroll-Bytes (immer 0) */

 0xFFFE, 0x8001, /* Sprite-Daten */
 0x8000, 0x8001,
 0x8000, 0x8001,
 0x8000, 0x8001,

```

```

0x8000, 0x8001,
0,0 /* Reserviertes Feld */
);

```

Die Besitzer eines Amiga mit mehr als 512 KByte RAM müssen an dieser Stelle noch einen Zwischenschritt einlegen. Da die Custom-Chips des Amiga nur die unteren 512 KByte RAM ansprechen können, muß dafür gesorgt werden, daß die Sprite-Daten, sprich die SpriteImage-Structure, in eben diesen 512 KByte liegt. Dazu muß mit AllocMem Speicher zur Verfügung gestellt werden. Anschließend müssen die Sprite-Daten in diesen Bereich kopiert werden. Einfacher geschieht dies mittels eines »Schalters«, der beim Kompilieren gesetzt werden muß. Genauer erfahren Sie in Ihrem C-Handbuch.

Im Anschluß daran muß das Sprite mit GetSprite reserviert werden. Es ist aber noch nicht auf dem Bildschirm zu sehen. Erst wenn das Sprite mit MoveSprite an seine Position gebracht wird, wird es dargestellt. Ist trotzdem nichts zu sehen, so wurde wahrscheinlich vergessen, das Flag »SPRITES« in der NewWindow-Structure zu setzen, was aber auch nachträglich durch den Befehl ON\_SPRITE erreicht werden kann.

Im Normalfall ist es allerdings nicht nötig, diese Flags zu setzen, da sie durch den Mauszeiger schon gesetzt werden. Wie dieser abgestellt werden kann, wird später erläutert.

Mehr Informationen sind aus dem Demonstrationsprogramm ersichtlich.

### 5.1.1 ChangeSprite

**Syntax:** ChangeSprite(ViewPort,SpritePtr,SpriteImPtr);

**Funktion:** Ändert ein Hardwaresprite, das zuvor mit GetSprite initialisiert worden sein muß.

**Parameter:** ViewPort -> Zeiger auf die ViewPort-Structure eines Screens.  
 SpritePtr -> Zeiger auf die SimpleSprite-Structure.  
 SpriteImPtr -> Zeiger auf die SpriteImage-Structure.

**Ergebnis:** Kein Ergebnis.

**Datentyp:** struct ViewPort \*ViewPort;  
 struct SimpleSprite \*SpritePtr;  
 struct SpriteImage \*SpriteImPtr;

**Sonstiges:** Auf den ViewPort kann man auf verschiedene Arten zugreifen:

Vom Window: WindowPtr -> WScreen -> ViewPort

Vom Screen: ScreenPtr -> ViewPort

Der ViewPort muß zu dem Screen gehören, auf dem die Sprites dargestellt werden sollen.

Der SpritePtr ist noch vom GetSprite-Befehl vorhanden.

Der SpriteImPtr muß neu erstellt werden, da er die neuen Daten für das Sprite enthält. Man kann hier auch die SpriteImage-Structure aus der SimpleSprite-Structure verwenden, auf die der SpritePtr zeigt.

Sie finden diesen Aufbau wahrscheinlich sehr kompliziert. In diesem Fall empfehlen wir Ihnen, die Beispiele genau zu betrachten. Wir sind sicher, daß Ihnen der Aufbau und die Logik dann um einiges klarer ist.

Ob für die Sprite-Daten nun  
struct SpriteImage \*SpriteImPtr;

oder

USHORT SpriteImPtr[];  
verwendet wird, ist unerheblich.

**Referenz:** Für die SimpleSprite- und die SpriteImage-Structure siehe auch Kapitel 5.1 »Einfache Hardware-Sprites«.

## 5.1.2 FreeSprite

**Syntax:** FreeSprite(SpriteNr);

**Funktion:** Löscht ein Hardwaresprite vom Screen.

**Parameter:** SpriteNr -> Hardwaresprite von 0 bis 7.

**Ergebnis:** Kein Ergebnis.

**Datentyp:** int SpriteNr;

**Sonstiges:** Dieser Befehl sollte immer dann angewendet werden, wenn ein Sprite nicht mehr benötigt wird, da das Amiga-System nur 8 Hardware-Sprites zur Verfügung stellt und somit nur allzu schnell alle Sprites belegt sind.

### 5.1.3 GetSprite

- Syntax:** `SpriteNr1 = GetSprite(SpritePtr,SpriteNr2);`
- Funktion:** Reserviert ein Hardware-Sprite.
- Parameter:** `SpritePtr` -> Zeiger auf die SimpleSprite-Structure.  
`SpriteNr2` -> Hardware-Sprite, das zu setzen ist.
- Ergebnis:** `SpriteNr1` -> Hardware-Sprite, das gesetzt wurde.
- Datentyp:** `struct SimpleSprite *SpritePtr;`  
`int SpriteNr2;`  
`int SpriteNr1;`
- Sonstiges:** Wird `SpriteNr2` gleich -1 gesetzt, wird das nächste freie Sprite benutzt und die entsprechende Nummer zurückgegeben. Wird für `SpriteNr2` eine Nummer von 0 bis 7 übergeben, so wird das entsprechende Sprite gesetzt und die Nummer wieder zurückgegeben.
- Ein Sonderfall tritt ein, wenn alle Sprites belegt sind, bzw. wenn das gewünschte Sprite besetzt ist. In diesem Fall wird der Wert -1 zurückgegeben und es wird kein Sprite gesetzt.
- Wenn das Sprite gesetzt wurde, wird die Nummer in die SimpleSprite-Structure eingetragen und zudem über `SpriteNr1` zurückgegeben.
- WICHTIG!** Dieser Befehl stellt das Sprite noch nicht auf dem Bildschirm dar. Erst wenn es mit `MoveSprite` positioniert worden ist, ist es zu sehen.
- Referenz:** Für die SimpleSprite-Structure siehe Kapitel 5.1 »Einfache Hardware-Sprites«.

### 5.1.4 MoveSprite

- Syntax:** `MoveSprite(ViewPort,SpritePtr,x,y);`
- Funktion:** Bewegt ein Hardware-Sprite zu einer spezifizierten Position und stellt es dort dar.
- Parameter:** `ViewPort` -> Zeiger auf die ViewPort-Structure eines Screens.  
`SpritePtr` -> Zeiger auf die SimpleSprite-Structure des Sprites, das bewegt werden soll.

x,y                    -> geben die neue Sprite-Position relativ zur linken, oberen Ecke des Screens an.

Ergebnis:            Kein Ergebnis.

Datentyp:            struct ViewPort \*ViewPort;  
                      struct SimpleSprite \*SpritePtr;  
                      int x, y;

Sonstiges:            Wie schon zuvor erwähnt, sind Sprites immer in der niedrigen Auflösung sichtbar. Das bedeutet, daß sie auf einem 640x400 Screen nur um jeweils zwei Punkte in jede Richtung bewegt werden können.

Wird dieser Befehl mit ChangeSprite kombiniert, so können schon einfache Trickfilme erstellt werden.

### 5.1.5 OFF\_SPRITE

Syntax:              OFF\_SPRITE();

Funktion:            Stellt den Sprite-DMA-Kanal ab.

Parameter:           Keine Parameter.

Ergebnis:           Kein Ergebnis.

Datentyp:            Keine Variablen.

Sonstiges:           Dieser Befehl bewirkt die Sperrung des Sprite-DMA-Kanals, was zur Folge hat, daß kein Sprite mehr sichtbar ist.

WICHTIG! OFF\_SPRITE ist ein Macro und muß speziell eingeladen werden. Siehe dazu den Anhang.

Referenz:            Siehe auch ON\_SPRITE

### 5.1.6 ON\_SPRITE

Syntax:              ON\_SPRITE();

Funktion:            Stellt den Sprite-DMA-Kanal an.

Parameter:           Keine Parameter.

Ergebnis:           Kein Ergebnis.

Datentyp:            Keine Variablen.

**Sonstiges:** Dieser Befehl bewirkt das Einschalten des Sprite-DMA-Kanals, was zur Folge hat, daß alle Sprites, die zuvor mit GetSprite initialisiert worden sind, sichtbar werden. Im Normalfall muß dieser Befehl nicht verwendet werden.

**WICHTIG!** ON\_SPRITE ist ein Macro und muß speziell eingeladen werden. Siehe dazu den Anhang.

**Referenz:** Siehe auch OFF\_SPRITE

```
1 /*****
2
3 Sprite / Demonstration
4 last update 26/06/87
5 von Joerg Koch und Frank Kremser
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Sprite-Demo laesst 4 bunte Smiles ueber den Screen flitzen, die sich
11 in der mitte >Guten Tag< sagen.
12
13 *****/
14
15 #include <exec/types.h> /* Include-Files die wir brauchen */
16 #include <exec/tasks.h>
17 #include <exec/libraries.h>
18 #include <exec/memory.h>
19 #include <exec/devices.h>
20 #include <devices/keymap.h>
21 #include <graphics/copper.h>
22 #include <graphics/display.h>
23 #include <graphics/gfxbase.h>
24 #include <graphics/text.h>
25 #include <graphics/view.h>
26 #include <graphics/gels.h>
27 #include <graphics/regions.h>
28 #include <graphics/sprite.h>
29 #include <hardware/blit.h>
30 #include <intuition/intuition.h>
31 #include <intuition/intuitionbase.h>
32
33
34 struct GfxBase *GfxBase; /* Lib Zeiger */
35 struct IntuitionBase *IntuitionBase;
36
37 struct Screen *screen; /* Screen -- Structure */
38
39 USHORT Data1[] =
40 (
41 0, 0,
42
43 0x0FC0, 0x0FC0,
44 0x3FF0, 0x3030,
45 0x7FFB, 0x400B,
46 0x7FFB, 0x400B,
47 0xF33C, 0xBCC4,
48 0xFFFC, 0xB004,
49 0xFFFC, 0xB004,
50 0xFCFC, 0xB304,
51 0xFFFC, 0xB004,
52 0xFFFC, 0x9024,
53 0x7FFB, 0x4B4B,
54 0x7FFB, 0x478B,
55 0x3FF0, 0x3030,
56 0x0FC0, 0x0FC0,
57
58 0,0
59);
60
61 USHORT Data2[] =
62 (
63 0, 0,
64
```

```

65 0x0FC0, 0x0FC0,
66 0x3FF0, 0x3030,
67 0x7FFB, 0x400B,
68 0x7FFB, 0x400B,
69 0xF33C, 0xBCC4,
70 0xFFFF, 0xB004,
71 0xFFFF, 0xB004,
72 0xFCFC, 0xB304,
73 0xFFFF, 0xB004,
74 0xFFFF, 0x9024,
75 0x7FFB, 0x4B4B,
76 0x7FFB, 0x47BB,
77 0x3FF0, 0x3030,
78 0x0FC0, 0x0FC0,
79
80 0,0
81 };
82
83 USHORT Data3[] =
84 (
85 0, 0,
86
87 0x0FC0, 0x0FC0,
88 0x3FF0, 0x3030,
89 0x7FFB, 0x400B,
90 0x7FFB, 0x400B,
91 0xF33C, 0xBCC4,
92 0xFFFF, 0xB004,
93 0xFFFF, 0xB004,
94 0xFCFC, 0xB304,
95 0xFFFF, 0xB004,
96 0xFFFF, 0x9024,
97 0x7FFB, 0x4B4B,
98 0x7FFB, 0x47BB,
99 0x3FF0, 0x3030,
100 0x0FC0, 0x0FC0,
101
102 0,0
103);
104
105 USHORT Data4[] =
106 (
107 0, 0,
108
109 0x0FC0, 0x0FC0,
110 0x3FF0, 0x3030,
111 0x7FFB, 0x400B,
112 0x7FFB, 0x400B,
113 0xF33C, 0xBCC4,
114 0xFFFF, 0xB004,
115 0xFFFF, 0xB004,
116 0xFCFC, 0xB304,
117 0xFFFF, 0xB004,
118 0xFFFF, 0x9024,
119 0x7FFB, 0x4B4B,
120 0x7FFB, 0x47BB,
121 0x3FF0, 0x3030,
122 0x0FC0, 0x0FC0,
123
124 0,0
125);
126
127 struct SimpleSprite spritel =
128 (
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

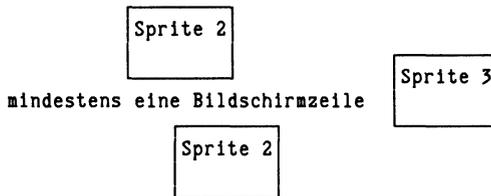
```
129 &Data1[0],
130 14,
131 100,
132 100,
133 2
134 };
135
136 struct SimpleSprite sprite2 =
137 {
138 &Data2[0],
139 14,
140 100,
141 100,
142 2
143 };
144
145 struct SimpleSprite sprite3 =
146 {
147 &Data3[0],
148 14,
149 100,
150 100,
151 2
152 };
153
154 struct SimpleSprite sprite4 =
155 {
156 &Data4[0],
157 14,
158 100,
159 100,
160 2
161 };
162
163 struct NewScreen ns =
164 {
165 0,
166 0,
167 320,
168 256,
169 2,
170 0,
171 1,
172 SPRITES,
173 CUSTOMSCREEN,
174 NULL,
175 NULL,
176 NULL,
177 NULL
178 };
179
180
181 main()
182 {
183 LONG warte;
184 USHORT schleife;
185
186 /* oeffnen der Libs */
187 if ((IntuitionBase = (struct IntuitionBase *)
188 OpenLibrary("intuition.library", 0)) == 0) exit();
189
190 if ((GfxBase = (struct GfxBase *)
191 OpenLibrary("graphics.library", 0)) == 0) exit();
192 /* oeffnen des Screens */
```

```
193 if ((screen = (struct Screen*) OpenScreen(&ns)) == NULL) exit();
194
195 SetRGB4(&screen->ViewPort,20,9,9,9); /* Farben setzen */
196 SetRGB4(&screen->ViewPort,21,11,11,11);
197 SetRGB4(&screen->ViewPort,22,13,13,13);
198 SetRGB4(&screen->ViewPort,23,15,15,15);
199 SetRGB4(&screen->ViewPort,24,15,0,0);
200 SetRGB4(&screen->ViewPort,25,0,15,0);
201 SetRGB4(&screen->ViewPort,26,0,0,15);
202 SetRGB4(&screen->ViewPort,27,15,0,15);
203
204 schleife = GetSprite(&sprite1,3); /* Sprite holen */
205 schleife = GetSprite(&sprite2,4);
206 schleife = GetSprite(&sprite3,5);
207 schleife = GetSprite(&sprite4,6);
208
209 /* Sprites ueber den Screen bewegen */
210 for (schleife = 0; schleife < 240; ++schleife)
211 {
212 for (warte = 0; warte < 1000; ++warte);
213 MoveSprite(&screen->ViewPort, &sprite1, schleife, schleife);
214 MoveSprite(&screen->ViewPort, &sprite2, 320-schleife, schleife);
215 MoveSprite(&screen->ViewPort, &sprite3, schleife, 256-schleife);
216 MoveSprite(&screen->ViewPort, &sprite4, 320-schleife, 256-schleife);
217 }
218 for(warte = 0; warte < 500000; ++warte);
219
220 FreeSprite(3); /* Sprites loeschen */
221 FreeSprite(4);
222 FreeSprite(5);
223 FreeSprite(6);
224
225 CloseScreen(screen); /* Screen und Libs */
226 CloseLibrary(GfxBase); /* schliessen */
227 CloseLibrary(IntuitionBase);
228 }
```

## 5.2 VSprites

VSprites, oder auch virtuelle Sprites, sind im Prinzip nicht anderes als Hardware-Sprites mit einer veränderten Handhabung. Ihr Vorteil liegt darin, daß sie die größte Schwäche von Hardware-Sprites umgehen: Es können mehr als nur 8 Sprites verwendet werden. Der Trick, der bei VSprites angewendet wird, um dies mit den Hardware-Sprites zu erreichen, besteht darin, daß Sprites mehrmals dargestellt werden und zwar mit verschiedenen Daten. Dazu muß aber zwischen der ersten und der zweiten Darstellung mindestens eine Bildschirmzeile liegen, da sonst die Hardware nicht mehr mitspielt.

Das Ganze sieht dann folgendermaßen aus:



Um das Handling braucht sich der Programmierer glücklicherweise nicht zu kümmern, dafür müssen bei Verwendung der VSprites aber einige Parameter mehr übergeben werden.

Für Breite, Höhe und Farbzahl gelten logischerweise die gleichen Beschränkungen, wie für Hardware-Sprites. Das heißt, sie sind 16 Punkte breit und bis zu 320 Punkte hoch, da Sprites immer in der niedrigen Auflösung gezeichnet werden. Sie dürfen maximal 4 Farben verwenden, wobei Farbe 1, wie bei den Hardware-Sprites, durchsichtig ist.

Hier ist aber ein weiterer Vorteil der VSprites gegenüber den Hardware-Sprites zu entdecken: jedes VSprite hat seine eigenen 3 Farben, in denen es dargestellt wird.

Folgendermaßen wird ein VSprite initialisiert:

Zuerst muß eine VSprite-Structure erstellt werden. Diese hat folgendes Aussehen:

```
struct VSprite{
 struct VSrite *NextVSprite; /* Nur für das System */
 struct VSrite *PrevVSprite;
 struct VSrite *DrawPath;
 struct VSrite *ClearPath;
 WORD OldY,OldX;
 WORD Flags; /* Folgende Flags sind möglich:
```

- VSPRITE:** Dieses Flag muß hier immer gesetzt werden, da wir an dieser Stelle nur VSprites verwenden. Nur wenn Bob's verwendet werden, ist dieses Flag nicht zu setzen.
- VSOVERFLOW:** Dieses Flag kann nicht vom Programmierer gesetzt werden, sondern nur gelesen werden. Ist es dann gesetzt, können nicht alle VSprites gezeigt werden, da sich zu viele überlappen.
- GELGONE:** Auch dieses Flag kann nur gelesen werden. Ist dieses Flag gesetzt, bedeutet dies, daß mindestens ein VSprite außerhalb des Screens positioniert ist.

```

WORD Y,X; /* VSprite-Position */
WORD Height; /* Höhe von 0 bis 256 Pixels */
WORD Width; /* wird ignoriert */
WORD Depth; /* wird ignoriert */
WORD MeMask; /* für Kollisionsabfrage */
WORD HitMask; /* für Kollisionsabfrage */
WORD *ImageData; /* Zeiger auf Spritedaten, die wie die Daten für
 ein Hardware-Sprite ermittelt werden */
WORD *BorderLine; /* für Kollisionsabfrage */
WORD *CollMask; /* für Kollisionsabfrage */
WORD *SprColors; /* VSprite-Farben */
struct Bob *VBob; /* wird ignoriert */
BYTE PlanePick; /* wird ignoriert */
BYTE PlaneOnOff; /* wird ignoriert */
VUserStuff VUserExt; /* für Benutzer-Anwendungen */ };
};

```

Viele der Parameter sind mit */\*wird ignoriert\*/* gekennzeichnet. Diese Parameter werden nur in Verbindung mit Bob's benötigt. Die Parameter für die Kollisionsabfrage lassen wir an dieser Stelle außer Acht, da dies ein weiterführendes Thema ist.

Wie das Structure initialisiert werden kann, ist aus der Demonstration ersichtlich.

Zu Beginn des Programms muß einmal das GEL-System (»Graphic Elements List«) installiert werden, wobei zum einen eine leere GelsInfo-Structure mit einigen Daten belegt werden muß. Welche belegt werden müssen, ist aus dem Demonstrationsprogramm ersichtlich. Anschließend muß sie mit InitGels installiert werden. Nach der Installation können dann die VSprites mit AddVSprite gesetzt werden. Ändern kann man sie, indem die VSprite-Structure geändert wird. Um das Sprite schließlich darzustellen, müssen folgende Befehle ausgeführt werden:

- ON\_DISPLAY und ON\_SPRITE um die DMA-Kanäle einzuschalten. (Ist im Normalfall nicht nötig)
- SortGList um die VSprites zu sortieren.
- DrawGList um die Liste bereitzustellen.
- MrgCop um die Liste dem Copper zur Verfügung zu stellen.
- LoadView um die Daten schließlich darzustellen.

Näheres ist aus den nachfolgenden Befehls Erläuterungen und dem Demonstrationsprogramm ersichtlich.

## 5.2.1 AddVSprite

Syntax: AddVSprite(Sprite,RastPort);

Funktion: Dieser Befehl fügt ein VSprite, das durch die VSprite-Structure festgelegt ist, in die VSprite-Liste ein.

Parameter: Sprite -> ist der Zeiger auf die VSprite-Structure des VSprites, das gezeigt werden soll.

RastPort -> ist ein Zeiger auf die RastPort-Structure des Screens oder Windows, das die VSprites kontrollieren soll.

Ergebnis: Kein Ergebnis.

Datentyp: struct VSprite \*Sprite;  
struct RastPort \*RastPort;

Sonstiges: Der Zeiger auf den RastPort ist der Window- oder der Screen-Structure zu entnehmen, was folgendermaßen geschieht:

Window: WindowPtr -> RPort

Screen: ScreenPtr -> RastPort

Die VSprite-Structure ist schon zuvor erläutert worden. Nähere Informationen über die Verwendung von VSprites entnehmen Sie bitte dem Demonstrationsprogramm.

## 5.2.2 DrawGLList

- Syntax:** DrawGLList(RastPort, ViewPort);
- Funktion:** Bereitet die VSprite-Liste für die Verwendung durch den Copper vor und stellt den benötigten Speicherplatz zur Verfügung
- Parameter:**
- |          |    |                                                                                                |
|----------|----|------------------------------------------------------------------------------------------------|
| RastPort | -> | Zeiger auf die RastPort-Structure, die die VSprites kontrolliert (siehe auch AddVSprite)       |
| ViewPort | -> | Zeiger auf die ViewPort-Structure des Screens, auf dem die VSprites dargestellt werden sollen. |
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct RastPort \*RastPort;  
struct ViewPort \*ViewPort;
- Sonstiges:** Dieser Befehl wird auch für die Verwendung von Bob's benötigt. In diesem Zusammenhang hat er noch andere Funktionen, doch da wir in diesem Kapitel nur auf VSprites eingehen wollen, lassen wir die zusätzlichen Funktionen außer Acht.

Wie auf den RastPort zugegriffen werden kann, ist unter AddVSprite erläutert. Auf den ViewPort kann man folgendermaßen zugreifen:

Vom Window: WindowPtr->WScreen->ViewPort

Vom Screen: ScreenPtr->ViewPort

### 5.2.3 InitGels

|            |                                                                                 |                                                                                             |
|------------|---------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| Syntax:    | InitGels(VSprite1, VSprite2, gelsinfo);                                         |                                                                                             |
| Funktion:  | Initialisiert eine Grafikelement-Liste, in die die VSprites eingetragen werden. |                                                                                             |
| Parameter: | VSprite1                                                                        | -> ist ein Zeiger auf eine leere VSprite-Structure, die den Anfang der Liste repräsentiert. |
|            | VSprite2                                                                        | -> ist ein Zeiger auf eine leere VSprite-Structure, die das Ende der Liste repräsentiert.   |
|            | gelsinfo                                                                        | -> ist ein Zeiger auf eine leere GelsInfo-Structure, die initialisiert werden soll.         |

Ergebnis: Kein Ergebnis.

Datentyp: struct VSprite \*VSprite1, \*VSprite2;  
struct GelsInfo \*gelsinfo;

Sonstiges: Wichtig! Dieser Befehl muß auf jeden Fall einmal aufgerufen werden, wenn VSprites benutzt werden sollen. Der Aufruf braucht nur einmal zu geschehen, muß vor dem ersten AddVSprite-Befehl stehen, weshalb er am Besten an den Anfang des Programms gesetzt wird.

Die VSprite-Structures, auf die VSprite1 und VSprite2 zeigen, sollten keine wichtigen Daten enthalten, da sie nur zur Erkennung des Beginns und des Endes der Liste dienen.

Die GelsInfo-Structure muß keine Werte enthalten, da diese von InitGels gesetzt werden. Folgendermaßen kann die GelsInfo-Structure deklariert werden:

```
struct GelsInfo *gelsinfo;
```

Nun kann gelsinfo, wie unter Syntax zu sehen ist, verwendet werden.

Da die GelsInfo-Structure keine besondere Bedeutung für den Programmierer hat, führen wir sie hier nicht auf. Sollten Sie trotzdem Interesse an dieser Structure haben, verweisen wir Sie auf den Anhang.

## 5.2.4 LoadView

|            |                                                                                                                                                                                                                                                                                                                                                                    |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | LoadView(view);                                                                                                                                                                                                                                                                                                                                                    |
| Funktion:  | Stellt auf dem Bildschirm die Informationen dar, wie sie in den Copper-Instruktionen festgelegt sind.                                                                                                                                                                                                                                                              |
| Parameter: | view           -> Zeiger auf eine View-Structure, die einen Zeiger auf die Copper-Instruktions-Liste besitzt.                                                                                                                                                                                                                                                      |
| Ergebnis:  | Kein Ergebnis.                                                                                                                                                                                                                                                                                                                                                     |
| Datentyp:  | struct View *view;                                                                                                                                                                                                                                                                                                                                                 |
| Sonstiges: | Dieser Befehl muß verwendet werden, um VSprites sichtbar zu machen, die mit AddVsprite in die VSprite-Liste eingetragen worden sind. Welche Befehle dazu noch zu verwenden sind, ersehen Sie aus Kapitel 5.2 »VSprites« und aus dem Demonstrations-Programm.<br><br>Auf die View-Structure kann folgendermaßen zugegriffen werden:<br><br>LoadView(ViewAddress()); |

## 5.2.5 MrgCop

|            |                                                                                                                              |
|------------|------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | MrgCop(view);                                                                                                                |
| Funktion:  | Trägt die nötigen Informationen, die der Copper benötigt, um die VSprites darzustellen, in die Copper-Instruktionsliste ein. |
| Parameter: | view           -> Zeiger auf eine View-Structure, in die neue Copper-Instruktionen eingefügt werden sollen.                  |
| Ergebnis:  | Kein Ergebnis.                                                                                                               |
| Datentyp:  | struct View *view;                                                                                                           |
| Sonstiges: | Auf die View-Structure kann folgendermaßen zugegriffen werden:<br><br>MrgCop(ViewAddress());                                 |

## 5.2.6 InitMasks

- Syntax:** InitMaks(VSprite);
- Funktion:** Dieser Befehl setzt die Umrandungen des VSprites und die dazugehörigen Kollisions-Masken.
- Parameter:** VSprite -> Zeiger auf die VSprite-Structure des VSprites, dessen Kollisions-Maske gesetzt werden soll.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct VSprite \*VSprite;
- Sonstiges:** Dieser Befehl erleichtert die Gestaltung der Kollisionsabfrage enorm. Er ermittelt die äußersten Punkte des VSprites und trägt die entsprechenden Daten in die VSprite-Structure ein.

## 5.2.7 RemVSprite

- Syntax:** RemVSprite(VSprite);
- Funktion:** Entfernt ein VSprite aus der Liste.
- Parameter:** VSprite -> Zeiger auf die VSprite-Structure des VSprites, das gelöscht werden soll.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct VSprite \*VSprite;
- Sonstiges:** Das VSprite kann anschließend einfach mit AddVSprite erneut in die Liste eingetragen werden.

## 5.2.8 SortGList

- Syntax:** SortGList(RastPort);
- Funktion:** Sortiert die VSprite-Liste.
- Parameter:** RastPort -> Zeiger auf die RastPort-Structure, die die VSprites kontrolliert.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct RastPort \*RastPort;

**Sonstiges:** Dieser Befehl sortiert die VSprite-Liste nach Y-Koordinaten. Das bedeutet, VSprites, die weiter oben erscheinen sollen, sind auch in der Liste weiter vorne zu finden. Dies ist nötig, um möglichst schnell zwischen zwei Bilddaten für die Hardware-Sprites, die ja zur Darstellung der VSprites verwendet werden, umschalten zu können.

Auf den RastPort kann man entweder über ein Window oder über einen Screen zugreifen. Dies geschieht folgendermaßen:

- Window: WindowPtr->RPort
- Screen: ScreenPtr->RastPort

## 5.2.9 WaitTOF

**Syntax:** WaitTOF();

**Funktion:** Wartet, bis der Elektronenstrahl der Bildröhre an der oberen Kante eines Screens angelangt ist.

**Parameter:** Keine Parameter.

**Ergebnis:** Kein Ergebnis.

**Datentyp:** Keine Variablen.

**Sonstiges:** Ist nur ein Screen auf dem Bildschirm zu sehen, wartet WaitTOF, bis der Elektronenstrahl am oberen Ende des Screens angelangt ist.

Dieser Befehl wird verwendet, um daß Programm mit dem Elektronenstrahl zu synchronisieren.

```
1 /*****
2
3 VSprite-Demonstration
4 last update 26/06/87
5 von Joerg Koch und Frank Kremser
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Diese Demonstration verwendet 12 VSprites, wovon eines quer ueber
11 den Bildschirm bewegt wird
12
13 *****/
14
15 #include <exec/types.h> /* Include-files laden */
16 #include <exec/tasks.h>
17 #include <exec/libraries.h>
18 #include <exec/exec.h>
19 #include <exec/devices.h>
20 #include <devices/keymap.h>
21 #include <graphics/copper.h>
22 #include <graphics/display.h>
23 #include <graphics/gfxbase.h>
24 #include <graphics/text.h>
25 #include <graphics/view.h>
26 #include <graphics/gels.h>
27 #include <graphics/regions.h>
28 #include <graphics/sprite.h>
29 #include <hardware/blit.h>
30 #include <intuition/intuition.h>
31 #include <intuition/intuitionbase.h>
32
33
34 struct GfxBase *GfxBase; /* Library-Zeiger */
35 struct IntuitionBase *IntuitionBase;
36
37 struct Screen *screen; /* Screen-Zeiger */
38 struct RastPort *rp; /* RastPort-Zeiger */
39
40 struct VSprite VS1, VS2; /* VSprite-Zeiger */
41
42 WORD spritecolors1[] = { 0x0000, 0x0888, 0x0FFF }; /* Spritefarben */
43 WORD spritecolors2[] = { 0x0900, 0x0800, 0x0F00 };
44 WORD spritecolors3[] = { 0x0090, 0x0040, 0x00F0 };
45 WORD spritecolors4[] = { 0x0009, 0x0383, 0x0FF0 };
46 WORD spritecolors5[] = { 0x0000, 0x0888, 0x0FBF };
47 WORD spritecolors6[] = { 0x0000, 0x0768, 0x07F0 };
48 WORD spritecolors7[] = { 0x0000, 0x0848, 0x0F0F };
49 WORD spritecolors8[] = { 0x0000, 0x0505, 0x045F };
50 WORD spritecolors9[] = { 0x0000, 0x0800, 0x006F };
51 WORD spritecolors10[] = { 0x0000, 0x0808, 0x0F30 };
52 WORD spritecolors11[] = { 0x0000, 0x0089, 0x00FB };
53 WORD spritecolors12[] = { 0x0000, 0x0208, 0x03F4 };
54
55 USHORT Data[] = /* Ein Spriteimage fuer alle Sprites */
56 {
57 0x0FC0, 0x0FC0,
58 0x3FF0, 0x3030,
59 0x7FFB, 0x4008,
60 0x7FFB, 0x4008,
61 0xF33C, 0xBCC4,
62 0xFFFF, 0x8004,
63 0xFFFF, 0x8004,
64 0xFCFC, 0x8304,
```

```
65 0xFFFC, 0x8004,
66 0xFFFC, 0x9024,
67 0x7FF8, 0x4848,
68 0x7FF8, 0x4788,
69 0x3FF0, 0x3030,
70 0x0FC0, 0x0FC0
71 };
72
73 struct VSprite sprite1 = /* Sprites deklarieren */
74 {
75 NULL,
76 NULL,
77 NULL,
78 NULL,
79 0,
80 0,
81 VSPRITE,
82 50,
83 50,
84 14,
85 14,
86 0,
87 0,
88 0,
89 &Data[0],
90 NULL,
91 NULL,
92 &spritecolors1[0],
93 NULL,
94 0,
95 0
96 };
97
98 struct VSprite sprite2 =
99 {
100 NULL,
101 NULL,
102 NULL,
103 NULL,
104 0,
105 0,
106 VSPRITE,
107 100,
108 100,
109 14,
110 14,
111 0,
112 0,
113 0,
114 &Data[0],
115 NULL,
116 NULL,
117 &spritecolors2[0],
118 NULL,
119 0,
120 0
121 };
122
123 struct VSprite sprite3 =
124 {
125 NULL,
126 NULL,
127 NULL,
128 NULL,
```

```
129 0,
130 0,
131 VSPRITE,
132 150,
133 150,
134 14,
135 14,
136 0,
137 0,
138 0,
139 &Data[0],
140 NULL,
141 NULL,
142 &spritecolors3[0],
143 NULL,
144 0,
145 0
146);
147
148 struct VSprite sprite4 =
149 {
150 NULL,
151 NULL,
152 NULL,
153 NULL,
154 0,
155 0,
156 VSPRITE,
157 100,
158 200,
159 14,
160 14,
161 0,
162 0,
163 0,
164 &Data[0],
165 NULL,
166 NULL,
167 &spritecolors4[0],
168 NULL,
169 0,
170 0
171);
172
173 struct VSprite sprite5 =
174 {
175 NULL,
176 NULL,
177 NULL,
178 NULL,
179 0,
180 0,
181 VSPRITE,
182 20,
183 10,
184 14,
185 14,
186 0,
187 0,
188 0,
189 &Data[0],
190 NULL,
191 NULL,
192 &spritecolors5[0],
```

```
193 NULL,
194 0,
195 0
196 };
197
198 struct VSprite sprite6 =
199 (
200 NULL,
201 NULL,
202 NULL,
203 NULL,
204 0,
205 0,
206 VSPRITE,
207 30,
208 300,
209 14,
210 14,
211 0,
212 0,
213 0,
214 &Data[0],
215 NULL,
216 NULL,
217 &spritecolors6[0],
218 NULL,
219 0,
220 0
221);
222
223 struct VSprite sprite7 =
224 (
225 NULL,
226 NULL,
227 NULL,
228 NULL,
229 0,
230 0,
231 VSPRITE,
232 190,
233 160,
234 14,
235 14,
236 0,
237 0,
238 0,
239 &Data[0],
240 NULL,
241 NULL,
242 &spritecolors7[0],
243 NULL,
244 0,
245 0
246);
247
248 struct VSprite sprite8 =
249 (
250 NULL,
251 NULL,
252 NULL,
253 NULL,
254 0,
255 0,
256 VSPRITE,
```

```
257 80,
258 70,
259 14,
260 14,
261 0,
262 0,
263 0,
264 &Data[0],
265 NULL,
266 NULL,
267 &spritecolors8[0],
268 NULL,
269 0,
270 0
271);
272
273 struct VSprite sprite9 =
274 {
275 NULL,
276 NULL,
277 NULL,
278 NULL,
279 0,
280 0,
281 VSPRITE,
282 200,
283 300,
284 14,
285 14,
286 0,
287 0,
288 0,
289 &Data[0],
290 NULL,
291 NULL,
292 &spritecolors9[0],
293 NULL,
294 0,
295 0
296);
297
298 struct VSprite sprite10 =
299 {
300 NULL,
301 NULL,
302 NULL,
303 NULL,
304 0,
305 0,
306 VSPRITE,
307 230,
308 10,
309 14,
310 14,
311 0,
312 0,
313 0,
314 &Data[0],
315 NULL,
316 NULL,
317 &spritecolors10[0],
318 NULL,
319 0,
320 0
```

```
321 };
322
323 struct VSprite sprite11 =
324 {
325 NULL,
326 NULL,
327 NULL,
328 NULL,
329 0,
330 0,
331 VSPRITE,
332 50,
333 160,
334 14,
335 14,
336 0,
337 0,
338 0,
339 &Data[0],
340 NULL,
341 NULL,
342 &spritecolors11[0],
343 NULL,
344 0,
345 0
346 };
347
348 struct VSprite sprite12 =
349 {
350 NULL,
351 NULL,
352 NULL,
353 NULL,
354 0,
355 0,
356 VSPRITE,
357 150,
358 240,
359 14,
360 14,
361 0,
362 0,
363 0,
364 &Data[0],
365 NULL,
366 NULL,
367 &spritecolors12[0],
368 NULL,
369 0,
370 0
371 };
372
373 struct NewScreen ns = /* Screen-Struktur */
374 {
375 0,
376 0,
377 320,
378 256,
379 2,
380 0,
381 1,
382 NULL,
383 CUSTOMSCREEN,
384 NULL,
```

```

385 NULL,
386 NULL,
387 NULL
388);
389
390
391 main()
392 {
393 LONG warte; /* Variable fuer Warte-Schleife */
394 struct GelsInfo gels; /* Leere GelsInfo-Structure */
395
396 if ((IntuitionBase = (struct IntuitionBase *) /* Libs oeffnen */
397 OpenLibrary("intuition.library", 0)) == 0) exit();
398
399 if ((GfxBase = (struct GfxBase *)
400 OpenLibrary("graphics.library", 0)) == 0) exit();
401
402 /* Screen oeffnen */
403 if ((screen = (struct Screen*) OpenScreen(&ns)) == NULL) exit();
404
405 /* RastPort zuweisen */
406 rp = &screen->RastPort;
407
408 gels.sprRsrvd = -1; /* GelsInfo deklarieren */
409 gels.nextLine = (WORD *)AllocMem(sizeof(WORD) * 8
410 ,MEMF_CLEAR|MEMF_PUBLIC);
411 gels.lastColor = (WORD **)AllocMem(sizeof(LONG) * 8
412 ,MEMF_CLEAR|MEMF_PUBLIC);
413 gels.collHandler = (struct collTable *)AllocMem(sizeof(struct
414 collTable),MEMF_CLEAR|MEMF_PUBLIC);
415 gels.leftmost = 0;
416 gels.rightmost = rp->BitMap->BytesPerRow * 8 - 1;
417 gels.topmost = 0;
418 gels.bottommost = rp->BitMap->Rows - 1;
419
420 rp->GelsInfo = &gels;
421
422 /* Gels initialisieren */
423 InitGels(&VS1,&VS2,&gels);
424
425 AddVSprite(&sprite1,rp); /* Sprites setzen */
426 AddVSprite(&sprite2,rp); /* 12 Sprites: */
427 AddVSprite(&sprite3,rp); /* Unterschied zu */
428 AddVSprite(&sprite4,rp); /* SimpleSprites */
429 AddVSprite(&sprite5,rp); /* wird somit */
430 AddVSprite(&sprite6,rp); /* deutlich, auch */
431 AddVSprite(&sprite7,rp); /* die Farbprioritaet */
432 AddVSprite(&sprite8,rp); /* der Hardwaresprite */
433 AddVSprite(&sprite9,rp); /* deutlich */
434 AddVSprite(&sprite10,rp);
435 AddVSprite(&sprite11,rp);
436 AddVSprite(&sprite12,rp);
437
438 for(warte = 0; warte < 230; warte ++)
439 {
440 sprite3.X = warte; /* ein Sprite bewegen */
441 sprite3.Y = warte;
442
443 SortGList(rp); /* VSprites neu sortieren */
444 DrawGList(rp,&screen->ViewPort);
445
446 MrgCop(ViewAddress()); /* in Copper-Liste eintragen */
447 LoadView(ViewAddress()); /* und darstellen. */
448 };

```

```
449
450 for(warte = 0; warte < 500000; ++warte);
451
452 RemVSprite(&sprite1); /* Sprites loeschen */
453 RemVSprite(&sprite2);
454 RemVSprite(&sprite3);
455 RemVSprite(&sprite4);
456 RemVSprite(&sprite5);
457 RemVSprite(&sprite6);
458 RemVSprite(&sprite7);
459 RemVSprite(&sprite8);
460 RemVSprite(&sprite9);
461 RemVSprite(&sprite10);
462 RemVSprite(&sprite11);
463 RemVSprite(&sprite12);
464
465 CloseScreen(screen); /* Libs und Screen schliessen */
466 CloseLibrary(GfxBase);
467 CloseLibrary(IntuitionBase);
468 }
```

## 5.3 Animation durch SetPointer

Intuition stellt dem Programmierer den Befehl `SetPointer` zur Verfügung, mit dessen Hilfe er einen windowspezifischen Mauszeiger verwenden kann. Dieser Zeiger ist immer dann sichtbar, wenn das betreffende Window aktiviert ist.

Für eigene Programme kann man sich also vorstellen, daß ein eigener Screen geöffnet wird, der von einem `Borderless-Window` ohne Gadgets abgedeckt wird. Das bedeutet, daß der entsprechende Screen nicht mehr mit der Maus »heruntergezogen« werden kann, was für einige Anwendungen sicherlich nicht wünschenswert wäre. Zudem ist die Screen-Titelzeile verdeckt, so daß ein völlig freier Bildschirm zu sehen ist. Nun kann der Programmierer einen eigenen Mauszeiger setzen, den er laufend durch einen neuen ersetzt. Somit kann mit sehr einfachen Mitteln Animation erzielt werden, ohne sich mit `Hardware-Sprites`, `VSprite-Structures` oder Ähnlichem herumschlagen zu müssen.

Der Befehl hat folgende Syntax:

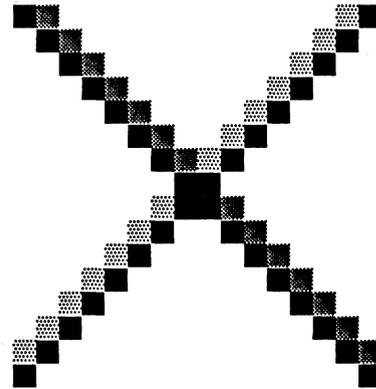
```
SetPointer(WindowPtr,Data,Höhe,Breite,X,Y);
```

»`WindowPtr`« ist ein Zeiger auf die `Window-Structure` des `Windows`, für das der Zeiger erstellt werden soll. »Höhe« und »Breite« geben Höhe und Breite des Mauszeigers in Pixel an, wobei diese in niedriger Auflösung, also 320 x 256 Pixel, angegeben werden müssen. Für die Breite des Zeigers gilt, wie für alle `Hardware-Sprites`, die Einschränkung auf 16 Pixel. `X` und `Y` geben die Position des sogenannten »Hot-Spot« an. Das ist der Punkt, der über einem `Gadget` oder einem Menüpunkt stehen muß, damit Intuition dies als gewählt erkennt. Seltsamerweise muß dieser Punkt in negativen Koordinaten, relativ zur linken, oberen Ecke des Zeigers angegeben werden. Das heißt, wenn der Zeiger die Form eines Kreuzes hat und 16 Punkte hoch wie breit ist, und den »Hot-Spot« in der Mitte haben soll, muß für `X` und `Y` jeweils -7 angegeben werden.

`Data` ist ein Zeiger auf die Daten für den neuen Mauszeiger. Wie schon im Kapitel über die einfachen `Hardware-Sprites` gesagt, muß auch bei dieser Anwendung darauf geachtet werden, daß die Daten innerhalb der unteren 512 KByte RAM liegen. Im Demonstrationsprogramm haben wir darauf verzichtet, da sich gezeigt hat, daß das Programm meistens auch mit mehr Speicherplatz einwandfrei lief. Sollte es dennoch Probleme geben, so verfahren Sie bitte nach dem gleichen Schema, wie bei den `Hardware-Sprites`.

Folgendermaßen müssen diese Daten bereitgestellt werden:

Nehmen wir an, daß für ein Window, mit dem Zeiger "WindowPtr" auf die zugehörige WindowStructure folgendes Kreuz als Mauszeiger gesetzt werden soll:



 bedeutet Farbe 1  
 bedeutet Farbe 2  
 bedeutet Farbe 3

Farbe 0, also durchsichtig, ist freigelassen.  
 Für Farbe 0 muß die erste und die zweite Bitplane frei sein.  
 Für Farbe 1 muß Bitplane 1 gesetzt und Plane 2 nicht gesetzt sein.  
 Für Farbe 2 muß Bitplane 2 gesetzt und Plane 1 nicht gesetzt sein.  
 Für Farbe 3 müssen beide Planes gesetzt sein.

Daraus folgt also:

|          |         |                  |          |
|----------|---------|------------------|----------|
| Zeile 1  | Plane 1 | 1000000000000011 | = 0x8003 |
|          | Plane 2 | 1100000000000001 | = 0xC001 |
| Zeile 2  | Plane 1 | 0100000000000110 | = 0x4006 |
|          | Plane 2 | 0110000000000010 | = 0x6002 |
| Zeile 3  | Plane 1 | 0010000000001100 | = 0x200C |
|          | Plane 2 | 0011000000000100 | = 0x3004 |
| Zeile 4  | Plane 1 | 0001000000011000 | = 0x1018 |
|          | Plane 2 | 0001100000001000 | = 0x1808 |
| Zeile 5  | Plane 1 | 0000100000110000 | = 0x0830 |
|          | Plane 2 | 0000110000010000 | = 0x0C10 |
| Zeile 6  | Plane 1 | 0000010001100000 | = 0x0460 |
|          | Plane 2 | 0000011000100000 | = 0x0620 |
| Zeile 7  | Plane 1 | 0000001011000000 | = 0x02C0 |
|          | Plane 2 | 0000001101000000 | = 0x0340 |
| Zeile 8  | Plane 1 | 0000000110000000 | = 0x0180 |
|          | Plane 2 | 0000000110000000 | = 0x0180 |
| Zeile 9  | Plane 1 | 0000000111000000 | = 0x0380 |
|          | Plane 2 | 0000000111000000 | = 0x01C0 |
| Zeile 10 | Plane 1 | 0000011001000000 | = 0x0640 |
|          | Plane 2 | 0000001001100000 | = 0x0260 |
| Zeile 11 | Plane 1 | 0000110000100000 | = 0x0C20 |
|          | Plane 2 | 0000010000110000 | = 0x0430 |

```

Zeile 12 Plane 1 0001100000010000 = 0x1810
 Plane 2 0000100000011000 = 0x0818
Zeile 13 Plane 1 0011000000010000 = 0x3008
 Plane 2 0001000000001100 = 0x100C
Zeile 14 Plane 1 0110000000000100 = 0x6004
 Plane 2 0010000000000110 = 0x4006
Zeile 15 Plane 1 1100000000000100 = 0xC002
 Plane 2 0100000000000111 = 0x4003
Zeile 16 Plane 1 1000000000000001 = 0x8001
 Plane 2 1000000000000001 = 0x8001

```

Im Programm sieht das dann folgendermaßen aus:

```

USHORT data[] =
{
 0x0000, 0x0000, /*Start-Bytes müssen immer 0 sein*/

 0x8003, 0xC001, /* Sprite-Daten von oben */
 0x4006, 0x6002,
 0x200C, 0x3004,
 0x1018, 0x1808,
 0x0830, 0x0C10,
 0x0460, 0x0620,
 0x02C0, 0x0340,
 0x0180, 0x0180,
 0x0380, 0x01C0,
 0x0640, 0x0260,
 0x0C20, 0x0430,
 0x1810, 0x0818,
 0x3008, 0x100C,
 0x6004, 0x4006,
 0xC002, 0x4003,
 0x8001, 0x8001,

 0x0000, 0x0000 /* End-Bytes müssen immer 0 sein*/
};

main()
{
 SetPointer(WindowPtr, &data[0], 16, 16, -7, -7);
}

```

Soll wieder der normale Intuition-Mauszeiger erscheinen, so kann dies mit `ClearPointer(WindowPtr)` erreicht werden.

Für dieses Thema siehe auch Kapitel 3.5.11 und 3.5.3.

Weitere Informationen können Sie aus dem folgenden Programm ableiten:

```

1 /*****
2
3 Pointer - Demonstration
4 last update 26/06/87
5 von Frank Kremser und Joerg Koch
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Pointer-Demonstration zeigt wie einfach die Veraenderung des Pointers
11 von statten geht.
12
13 *****/
14
15 #include <exec/types.h> /* Include / Files */
16 #include <exec/nodes.h>
17 #include <exec/lists.h>
18 #include <exec/ports.h>
19 #include <exec/memory.h>
20 #include <exec/devices.h>
21 #include <devices/keymap.h>
22 #include <graphics/regions.h>
23 #include <graphics/copper.h>
24 #include <graphics/gels.h>
25 #include <graphics/gfxbase.h>
26 #include <graphics/gfx.h>
27 #include <graphics/clip.h>
28 #include <graphics/view.h>
29 #include <graphics/rastport.h>
30 #include <graphics/layers.h>
31 #include <intuition/intuition.h>
32 #include <hardware/blit.h>
33
34 struct IntuitionBase *IntuitionBase; /* Lib - Zeiger */
35 struct GfxBase *GfxBase;
36
37 struct RastPort *rp;
38 struct Window *window;
39
40 struct NewWindow nw = /* New Window - Structure */
41 (
42 0, /* linke Ecke */
43 0, /* obere Ecke */
44 640, /* Breite */
45 256, /* Hoehe */
46 3, /* DetailPen */
47 1, /* BlockPen */
48 NULL, /* IXCMP / Flags */
49 ACTIVATE|NOICARE|FRESH, /* Flags */
50 NULL, /* Erstes Gadget */
51 NULL, /* CheckMark */
52 "Animation durch SetPointer", /* Window Titel */
53 NULL, /* Pointer auf den screen */
54 NULL, /* Pointer auf superbitmap */
55 0, /* min Breite */
56 0, /* min Hoehe */
57 0, /* max Breite */
58 0, /* max Hoehe */
59 WBENCHSCREEN /* Typ */
60);
61
62 USHORT Image1[] = /* PointerImage */
63 (
64 0,0,

```

```
65
66 0x03C0, 0x0000,
67 0x0C30, 0x0000,
68 0x100B, 0x0000,
69 0x23C4, 0x03C0,
70 0x4422, 0x0420,
71 0x4B12, 0x0B10,
72 0x9009, 0x100B,
73 0x9009, 0x100B,
74 0x9009, 0x100B,
75 0x9009, 0x100B,
76 0x4B12, 0x0B10,
77 0x4422, 0x0420,
78 0x23C4, 0x03C0,
79 0x100B, 0x0000,
80 0x0C30, 0x0000,
81 0x03C0, 0x0000,
82
83 0,0
84);
85
86 USHORT Image2[] = /* PointerImage */
87 (
88 0,0,
89
90 0x03C0, 0x0000,
91 0x0C30, 0x0000,
92 0x100B, 0x0000,
93 0x2004, 0x0000,
94 0x41B2, 0x01B0,
95 0x4242, 0x0240,
96 0xB421, 0x0420,
97 0xBB11, 0x0B10,
98 0xBB11, 0x0B10,
99 0xB421, 0x0420,
100 0x4242, 0x0240,
101 0x41B2, 0x01B0,
102 0x2004, 0x0000,
103 0x100B, 0x0000,
104 0x0C30, 0x0000,
105 0x03C0, 0x0000,
106
107 0,0
108);
109
110 USHORT Image3[] = /* PointerImage */
111 (
112 0,0,
113
114 0x03C0, 0x0000,
115 0x0C30, 0x0000,
116 0x100B, 0x0000,
117 0x2004, 0x0000,
118 0x4002, 0x0000,
119 0x4002, 0x0000,
120 0xB1B1, 0x01B0,
121 0xB241, 0x0240,
122 0xB241, 0x0240,
123 0xB1B1, 0x01B0,
124 0x4002, 0x0000,
125 0x4002, 0x0000,
126 0x2004, 0x0000,
127 0x100B, 0x0000,
128 0x0C30, 0x0000,
```

```
129 0x03C0, 0x0000,
130
131 0,0
132 };
133
134 main()
135 {
136 LONG schleife;
137 LONG warte; /* oeffnen der Libs */
138
139 if(!(GfxBase = (struct GfxBase *)OpenLibrary("graphics.library",0)))
140 {
141 close_things();
142 exit();
143 };
144
145 if(!(IntuitionBase = (struct IntuitionBase *)
146 OpenLibrary("intuition.library",0)))
147 {
148 close_things();
149 exit();
150 }; /* oeffnen des Window */
151
152 if (!(window = (struct Window *)OpenWindow(&nw)))
153 {
154 close_things();
155 exit();
156 };
157
158 /* abwechselndes Setzen der Pointer */
159 for(schleife = 0; schleife < 50; schleife++)
160 {
161 SetPointer(window,&Image1[0],16,16,-7,7);
162 for(warte = 0; warte < 10000; warte++);
163 SetPointer(window,&Image2[0],16,16,-7,7);
164 for(warte = 0; warte < 10000; warte++);
165 SetPointer(window,&Image3[0],16,16,-7,7);
166 for(warte = 0; warte < 10000; warte++);
167 SetPointer(window,&Image2[0],16,16,-7,7);
168 for(warte = 0; warte < 10000; warte++);
169 };
170 close_things(); /* Unterroutine zum Schliessen */
171 }
172
173 close_things()
174 {
175 CloseWindow(window); /* schliessen des Window */
176 CloseLibrary(GfxBase); /* und Libs */
177 CloseLibrary(IntuitionBase);
178 }
```

## 5.4 Animation durch Preferences

Die Befehle funktionieren sowohl mit der alten Workbench-Version 1.1 als auch mit der Version 1.2.

Auch mit Preferences können Effekte erzielt werden. Ein solcher Effekt ist zum Beispiel das Verschieben des Bildschirmausschnittes. Aber auch die anderen Voreinstellungen, die mit dem Programm Preferences eingestellt werden können, sind veränderbar, wie zu sehen sein wird.

Um auf die Preferences-Daten zugreifen zu können, sind zwei Befehle vorhanden:

- GetDefPrefs
- GetPrefs

Der Unterschied zwischen den beiden Befehlen wird später erläutert. Als Parameter verlangen beide Befehle einen Zeiger auf einen Speicherbereich, in den die Preference-Daten kopiert werden sollen und die Byteanzahl, die kopiert werden soll. Dies ist zwar umständlicher als wenn direkt eine Preference-Structure angegeben werden würde, hat aber den Vorteil, daß ein Programm, das auf diese Weise zu den erwünschten Daten gelangt, auch auf die Nachfolgeversionen des Amiga übertragbar ist.

Üblicherweise verwendet man den Befehl in folgender Weise:

```
.
. struct Preferences pref;
.
. main()
{
.
. GetPrefs(&pref, sizeof(struct Preferences));
.
. }
}
```

Im Anschluß an den Befehl GetPrefs oder GetDefPrefs können die Daten beliebig manipuliert werden. Wie, ist aus dem Demonstrationsprogramm ersichtlich.

Um die Daten dann auch verwenden zu können, müssen sie mit SetPrefs entweder auf Diskette gesichert werden, von wo sie nach jedem Bootvorgang geladen werden oder in den Speicher zurückkopiert werden, wobei sie sofort vom System verwendet werden.

Welche Daten modifiziert werden können, ist aus der Preference-Structure ersichtlich:

```

struct Preferences
{
 BYTE FontHeight;
 USHORT PrinterPort;
 USHORT BaudRate;
 struct timeval KeyRptSpeed, KeyRptDelay;
 struct timeval DoubleClick;
 USHORT PointerMatrix[POINTERSIZE];
 BYTE XOffset, YOffset;
 USHORT color17, color18, color19;
 USHORT PointerTicks;
 USHORT color0, color1, color2, color3;
 BYTE ViewXOffset, ViewYOffset;
 WORD ViewInitX, ViewInitY;
 BOOL EnableCLI;
 USHORT PrinterType;
 UBYTE PrinterFilename[FILENAME_SIZE];
 USHORT PrintPitch;
 USHORT PrintQuality;
 USHORT PrintSpacing;
 WORD PrintLeftMargin, PrintRightMargin;
 USHORT PrintImage;
 USHORT PrintAspect;
 USHORT PrintShade;
 WORD PrintThreshold;
 USHORT PaperSize;
 WORD PaperLength;
 USHORT PaperType;
 UBYTE SerRWBits;
 UBYTE SerStopBut;
 UBYTE SerParShk;
 UBYTE LaceWB;
 UBYTE WorkName[FILE];
 BYTE RowSizeChange;
 BYTE ColumnSizeChange;
 BYTE Padding[14];
};

```

### 5.4.1 GetDefPrefs

**Syntax:** GetDefPrefs(PrefBuffer,Bytes);

**Funktion:** Holt die Preference-Einstellungen, die beim Booten eingestellt waren von Diskette.

**Parameter:** PrefBuffer -> Zeiger auf den Speicherbereich, in den die Preference-Daten kopiert werden sollen.

Bytes -> Anzahl der Bytes, die kopiert werden sollen.

**Ergebnis:** Kein Ergebnis.  
**Datentyp:** ULONG PrefBuffer;  
int Bytes;  
**Sonstiges:** Wie die Parameter am besten gewählt werden, ist unter Kapitel 5.3 »Animation mit Preferences« beschrieben.  
**Referenz:** Siehe auch GetPrefs

## 5.4.2 GetPrefs

**Syntax:** GetPrefs(PrefBuffer,Bytes);  
**Funktion:** Holt die Preference-Einstellungen, die momentan eingestellt sind, aus dem Speicher.  
**Parameter:** PrefBuffer -> Zeiger auf den Speicherbereich, in den die Preference-Daten kopiert werden sollen  
Bytes -> Anzahl der Bytes, die kopiert werden sollen  
**Ergebnis:** Kein Ergebnis.  
**Datentyp:** ULONG PrefBuffer;  
int Bytes;  
**Sonstiges:** Wie die Parameter am besten gewählt werden, ist unter Kapitel 5.3 »Animation mit Preferences« beschrieben.  
**Referenz:** Siehe auch GetDefPrefs

## 5.4.3 SetPrefs

**Syntax:** SetPrefs(PrefBuffer,Bytes,Disk);  
**Funktion:** Speichert die Preference-Einstellungen entweder auf Diskette oder in den dafür vorgesehenen Speicherbereich.  
**Parameter:** PrefBuffer -> Zeiger auf den Speicherbereich, der die Preference-Daten enthält, die gesichert werden sollen.  
Bytes -> Anzahl der Bytes, die gesichert werden sollen.

Disk -> ist eine Boolesche Variable und setzt fest, ob die veränderten Preference-Daten dauerhaft oder nur vorübergehend gespeichert werden sollen.

**Ergebnis:** Kein Ergebnis.

**Datentyp:** ULONG PrefBuffer;  
int Bytes;  
bool Disk;

**Sonstiges:** Wie PrefBuffer und Bytes am besten gewählt werden, ist unter Kapitel 5.4 »Animation mit Preference« beschrieben.  
Dieser Befehl sollte nur in Ausnahmesituationen verwendet werden, da er unter Umständen fatale Folgen haben kann.

**Referenz:** Siehe auch GetDefPrefs und GetPrefs1

```
1 /*****
2
3 Preferences-Demonstration
4 last update 26/05/87
5 Frank Kremser und Joerg Koch
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Zeigt die Leistungsfaehigkeit von Preferences !!!!
11 Durch verschieben der Offsets kann Prefs auch zur Animation verwendet
12 werden.
13
14 *****/
15
16 #include <exec/types.h> /* Include-Files */
17 #include <exec/nodes.h>
18 #include <exec/lists.h>
19 #include <exec/ports.h>
20 #include <exec/devices.h>
21 #include <devices/keymap.h>
22 #include <graphics/regions.h>
23 #include <graphics/copper.h>
24 #include <graphics/gels.h>
25 #include <graphics/gfxbase.h>
26 #include <graphics/gfx.h>
27 #include <graphics/clip.h>
28 #include <graphics/view.h>
29 #include <graphics/rastport.h>
30 #include <graphics/layers.h>
31 #include <intuition/intuition.h>
32 #include <hardware/blit.h>
33
34 struct IntuitionBase *IntuitionBase; /* Lib-Zeiger */
35 struct GfxBase *GfxBase;
36
37
38 main()
39 {
40 struct Preferences pref; /* Preferences-Structure bereitstellen */
41 LONG schleife; /* Variablen fuer Animation */
42 LONG x,y;
43 /* Oeffnen der Libs */
44 if(!(GfxBase = (struct GfxBase *)
45 OpenLibrary("graphics.library",0))) exit();
46
47 if(!(IntuitionBase = (struct IntuitionBase *)
48 OpenLibrary("intuition.library",0))) exit();
49
50 GetPrefs(&pref,sizeof(struct Preferences)); /* holen von Preferences */
51 x = pref.ViewXOffset;
52 y = pref.ViewYOffset; /* Offset retten */
53 pref.ViewYOffset = 0;
54 SetPrefs(&pref,sizeof(struct Preferences),FALSE); /* neue */
55 /* Prefs setzen */
56
57 for(schleife = 0; schleife < 255; schleife++) /* Offset laufend */
58 /* veraendern */
59 {
60 pref.ViewXOffset = schleife;
61 SetPrefs(&pref,sizeof(struct Preferences),FALSE); /* Prefs setzen */
62 };
63
64 for(schleife = 0; schleife < 255; schleife++)
```

```
65 {
66 pref.ViewYOffset = schleife;
67 SetPrefs(&pref,sizeof(struct Preferences),FALSE); /* Prefs setzen */
68 };
69
70 for(schleife = 255; schleife > 0; schleife--)
71 {
72 pref.ViewXOffset = schleife;
73 SetPrefs(&pref,sizeof(struct Preferences),FALSE); /* Prefs setzen */
74 };
75
76 for(schleife = 255; schleife > 0; schleife--)
77 {
78 pref.ViewYOffset = schleife;
79 SetPrefs(&pref,sizeof(struct Preferences),FALSE); /* Prefs setzen */
80 };
81
82 pref.ViewXOffset = x; /* alten Offset setzen */
83 pref.ViewYOffset = y;
84 SetPrefs(&pref,sizeof(struct Preferences),FALSE);
85
86 CloseLibrary(GfxBase); /* Libs schliessen */
87 CloseLibrary(IntuitionBase);
88 }
```



# Die Programmbedienung

Die neue Rechnergeneration, die derzeit auch durch die Amiga-Serie repräsentiert wird, stellt nicht nur mehr (Rechen-) Leistung zur Verfügung, sondern besitzt auch neue Betriebssysteme, die sich besonders durch ihre grafische Benutzeroberfläche, beim Amiga »Intuition« genannt, hervorheben. Diese Benutzeroberflächen erleichtern dem ungeübten Benutzer das Arbeiten auf dem Computer ungemein.

Für professionelle Programme auf solchen Rechnern ist die Verwendung dieses Hilfsmittels praktisch schon ein Muß, auch wenn es ab und an kritisiert wird.

Die Vorteile dieses Systems liegen auf der Hand:

- Einfache Handhabung
- Schnelle Einarbeitung
- Kurze Lernphase

und damit hohe Bedienerfreundlichkeit.

Natürlich hat dieses System bei einigen Anwendungen auch Nachteile, doch können diese ausgemerzt werden, indem in diesen Fällen die konventionellen Methoden verwendet werden, die das Amiga-System in Form von CLI zur Verfügung stellt.

## 6.1 Programmbedienung mit dem Amiga

In diesem Kapitel wollen wir uns mit der Bedienung eines Programms durch den Benutzer beschäftigen. Dem Programmierer stehen dabei verschiedene Mittel zur Verfügung, mit deren Hilfe er die Bedienung für den Benutzer so komfortabel wie möglich gestalten kann:

- Menüs  
Menüs kann der Benutzer erreichen, indem er die Menütaste – die rechte Maustaste – betätigt, auf einen Menüoberbegriff geht, wodurch die verschiedenen Menüunterpunkte sichtbar werden, und sich dann einen dieser Punkte auswählt. Dabei kann noch ein weiteres Untermenü sichtbar werden.

Menüpunkte können entweder Aktionen sein, die das Programm ausführt, oder aber Schalter, die gesetzt werden können. Ist ein Menüschalter gesetzt, wird ein »Checkmark« (siehe auch Kapitel 3 »Das Window« – Window-Structure) daneben gezeichnet.

- Gadgets  
Intuition stellt eine Vielzahl von Gadget-Typen zur Verfügung, die vom Programmierer verwendet werden können. Im einzelnen sind sie in Kapitel 8 erläutert.

Sie können als Schalter, die betätigt werden müssen, oder aber zur Eingabe dienen. Auch können Schiebepotentiometer mit Hilfe von Proportionalgadgets realisiert werden. Solche werden zum Beispiel zur Einstellung von Farben verwendet.

Die zuvor genannten Möglichkeiten werden zur Eingabe verwendet.

Aber auch zur Ausgabe stehen Mittel zur Verfügung:

- Alerts  
Alerts sollten nur dann verwendet werden, wenn dem Benutzer wirklich wichtige Meldungen zu übermitteln sind, da sie eine »Schock«-Behandlung darstellen. Es können zwei Alert-Typen verwendet werden. Einmal die System-Alerts, die schon »vorgefertigt« sind, also nicht verändert werden können. Die zweite Möglichkeit ist die Verwendung von eigenen Alerts. Bei diesen kann der Programmierer einen eigenen Text angeben.

– Requester

Requester können ausgegeben werden, um den Programmbenutzer zu einer bestimmten Handlung zu bewegen, ein Beispiel dafür ist der »No Disk Present«-Requester, oder um bestimmte Eingaben abzufragen, wie der »Load File«-Requester bei Graphicraft.



## Die Menüs

Intuition stellt dem Programmierer sogenannte Menüs zur Verfügung. Dies sind Kästen, die erscheinen, wenn der rechte Mausknopf betätigt wird und der Mauszeiger über einem Menüpunkt steht. Anschließend kann einer der Punkte, die in dem Kasten aufgeführt sind, angewählt werden. Die Punkte können entweder als Grafik oder als Text dargestellt werden. Zudem kann ein Menüpunkt entweder eine Handlung starten oder aber einen Zustand setzen, also ein Attribut.

## 7.1 Der Aufbau der Menüs

Ist die rechte Maustaste gedrückt, so erscheint am oberen Screen-Rand die Menüzeile. Diese Menüzeile enthält verschiedene Menüoberbegriffe. Steht die Maus auf einem Menüoberbegriff, so erscheint der zugehörige Menükasten. Dieser Kasten enthält die verschiedenen Einträge, Items genannt. Bei verschiedenen Items erscheint ein weiterer Kasten, Subitem genannt.

Jedes dieser Items, bzw. Subitems kann bei Betätigung entweder ein Attribut setzen oder aber eine Handlung starten. Das CleanUp-Item auf der Workbench startet zum Beispiel eine Handlung. Dahingegen sind die Farbitems bei der Farbauswahl in Graphicraft Items, die Attribute setzen, nämlich die Farbe. Normalerweise werden an Menüpunkte, die Attribute setzen, eine »Checkmark« – ein Häkchen – gezeichnet. Soll ein anderes Zeichen erscheinen, so kann dieses Zeichen in der zugehörigen Window-Structure gesetzt werden (siehe dort).

Wir wollen nun ein Menü aufbauen. Dieses Menü soll nur einen Eintrag – ein Item – haben. Dazu müssen wir als erstes den Text setzen, den dieses Item haben soll. Dies geschieht über IntuiText:

```
struct IntuiText itemtext =
{
 2,
 0,
 JAM1,
 3,
 3,
 NULL,
 "Test-Item",
 NULL
};
```

Anschließend muß das Item selbst erstellt werden. Dazu benötigen wir eine MenuItem-Structure. Diese hat folgende Form:

```
struct MenuItem
{
 struct MenuItem *NextItem; Zeiger auf nächstes Item im Menü
 USHORT LeftEdge, TopEdge; linke, obere Ecke des Items
 USHORT Width, Height; Breite und Höhe des Items
 USHORT Flags; Item-Spezifikationen
 LONG MutualExclude; Exklusiv-Bedingung
 APTR ItemFill; Zeiger auf Image oder IntuiText
 APTR SelectFill; Zeiger auf alternativ Image o. Text
 BYTE Command; Command-Zeichen
 struct MenuItem *SubItem; Zeiger auf zugehöriges Subitem
 USHORT NextSelect; benötigt Intuition
}
```

MutualExclude, Command und Subitem werden später erklärt.

Für Flags stehen folgende Möglichkeiten zur Verfügung:

|             |                                                                                                                                                                                                      |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CHECKIT     | Ist dieses Flag gesetzt, setzt das zugehörige Item ein Attribut.                                                                                                                                     |
| CHECKED     | Ist dieses Flag gesetzt, bedeutet dies, daß das Attribut zu Beginn gesetzt ist. Es wird also gleich zu Anfang ein Häkchen gezeichnet.                                                                |
| ITEMTEXT    | Das Item ist ein Text-Item.                                                                                                                                                                          |
| COMMSEQ     | Zu diesem Item gibt es eine Tasten-Sequenz, die die gleiche Funktion hat. Zudem wird die Tastensequenz mit in das Menü gezeichnet.                                                                   |
| ITEMENABLED | Dieses Flag beschreibt, ob das Item eingeschaltet ist, das heißt, ob es abgefragt wird. Ist es nicht eingeschaltet, wird es gepunktet dargestellt.                                                   |
| HIGHCOMP    | Das Item wird komplementiert dargestellt, wenn der Mauszeiger über ihm steht.                                                                                                                        |
| HIGHBOX     | Das Item wird umrahmt, wenn der Mauszeiger über ihm steht.                                                                                                                                           |
| HIGHIMAGE   | Das Item wird nicht durch das normale Image, bzw. den Text, sondern durch das Image, bzw. den Text dargestellt, auf das, bzw. den durch SelectFill gezeigt wird, wenn der Mauszeiger über ihm steht. |
| HIGHNONE    | Das Item wird nicht verändert, wenn der Mauszeiger über ihm steht.                                                                                                                                   |
| ISDRAW      | Dieses Flag wird von Intuition gesetzt, wenn das Item zur Zeit zu sehen ist.                                                                                                                         |
| HIGHITEM    | Dieses Flag wird von Intuition gesetzt, wenn der Mauszeiger über diesem Item steht.                                                                                                                  |

Unser Item hat also folgende Form:

```
struct MenuItem item =
{
 NULL, /* Kein weiteres Item im Menü */
 2,
 2,
 98,
 12,
 ITEMTEXT|ITEMENABLED|HIGHBOX,
 0,
 (APTR) &itext, /* Zeiger auf unser Image
 NULL, /* Keine Alternativ-Image
 0,
```

```
 NULL,
 0
};
```

Anschließend muß noch die Menü-Structure für den Menüoberpunkt erstellt werden, unter dem unser Item erscheinen soll. Diese hat folgende Form:

```
struct Menu
{
 struct Menu*NextMenu; Zeigeraufdenächsten Menüpunkt
 SHORT LeftEdge, TopEdge; linke, obere Ecke
 SHORT Width, Height; Breite und Höhe des Menüoberpunkt
 USHORT Flags;
 BYTE *MenuName; Text des Menüoberpunktes
 struct MenuItem *FirstItem; Zeiger auf erstes Item des Menüs
};
```

Für Flags gibt es folgende Möglichkeiten:

|                    |                                                                                       |
|--------------------|---------------------------------------------------------------------------------------|
| <b>MENUENABLED</b> | Das Menü ist eingeschaltet. Wenn es ausgeschaltet ist, wird es gepunktet dargestellt. |
| <b>MIDDRAWN</b>    | Intuition setzt dieses Flag, wenn der zugehörige Menükasten zu sehen ist              |

Unsere Menu-Structure sieht also folgendermaßen aus:

```
struct Menu menu =
{
 NULL,
 0,
 0,
 100,
 10,
 MENUENABLED,
 "Test-Menu",
 &item
};
```

Nun können wir das Menü mit `SetMenuStrip ((windowPtr,&menu);` setzen. Immer wenn das Window, zu dem das Menü gehört, aktiviert ist, kann das Menü angewählt werden.

### 7.1.1 ClearMenuStrip

Syntax: `ClearMenuStrip(windowPtr);`

Funktion: Löscht alle Menüs, die zu dem spezifizierten Window gehören.

Parameter: `windowPtr` -> Zeiger auf die Window-Structure des Windows, dessen Menüs gelöscht werden sollen.

|            |                                                                                               |
|------------|-----------------------------------------------------------------------------------------------|
| Ergebnis:  | Kein Ergebnis.                                                                                |
| Datentyp:  | struct Window *windowPtr;                                                                     |
| Sonstiges: | Mit SetMenuStrip können anschließend neue oder aber wieder die gleichen Menüs gesetzt werden. |
| Referenz:  | Siehe auch SetMenuStrip                                                                       |

## 7.1.2 ItemAddress

|            |                                                                                                                                                                                                                                                                |                                                                          |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|
| Syntax:    | adr = ItemAddress(menunr,itemnr);                                                                                                                                                                                                                              |                                                                          |
| Funktion:  | Ermittelt die Adresse der MenuItem-Structure, die über die Menü-Nummer und Item-Nummer angegeben ist.                                                                                                                                                          |                                                                          |
| Parameter: | menunr                                                                                                                                                                                                                                                         | -> Nummer des Menüs, in dem sich das Item befindet.                      |
|            | itemnr                                                                                                                                                                                                                                                         | -> Nummer des Items, nach dessen MenuItem-Structure gesucht werden soll. |
| Ergebnis:  | adr                                                                                                                                                                                                                                                            | -> Adresse, an der sich die MenuItem-Structure befindet.                 |
| Datentyp:  | int menunr, itemnr;<br>ULONG adr;                                                                                                                                                                                                                              |                                                                          |
| Sonstiges: | Bei der Abfrage der Menüs erhält man die Nummern der Menüs zurück, unter denen diese von Intuition gespeichert werden. Diese Nummern können durch diesen Befehl in eine Adresse gewandelt werden, mit der man leichter erfährt, welches Item ausgewählt wurde. |                                                                          |

Das Ganze funktioniert allerdings nicht bei Subitems.

## 7.1.3 ITEMNUM

|            |                                                       |                                                                          |
|------------|-------------------------------------------------------|--------------------------------------------------------------------------|
| Syntax:    | nr = ITEMNUM(code);                                   |                                                                          |
| Funktion:  | Ermittelt die Nummer des Items, das ausgewählt wurde. |                                                                          |
| Parameter: | code                                                  | -> Code-Wert aus der Message-Structure der Message, die übergeben wurde. |
| Ergebnis:  | nr                                                    | -> Nummer des Items, das ausgewählt wurde.                               |

- Datentyp:** SHORT code;  
int nr;
- Sonstiges:** Die Items sind aufsteigend von 0 an numeriert.
- Referenz:** Siehe auch Kapitel 7.3 »Die Abfrage von Menüs«, MENUNUM, SUBNUM

### 7.1.4 MENUNUM

- Syntax:** nr = MENUNUM(code);
- Funktion:** Ermittelt die Nummer des Menüs, das angewählt wurde.
- Parameter:** code -> Code-Wert aus der Message-Structure der Message, die übergeben wurde.
- Ergebnis:** nr -> Nummer des Menüs, das angewählt wurde.
- Datentyp:** SHORT code;  
int nr;
- Sonstiges:** Die Menüs sind aufsteigend von 0 an numeriert.
- Referenz:** Siehe auch Kapitel 7.3 »Die Abfrage von Menüs«, ITEMNUM, SUBNUM

### 7.1.5 OffMenu

- Syntax:** OffMenu(windowptr,nr);
- Funktion:** Schaltet einen ganzen Menüoberpunkt ab.
- Parameter:** windowptr -> Zeiger auf das Window, zu dem das Menü gehört.  
nr -> Nummer des Menüs, das abgeschaltet werden soll.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct Window \*windowptr;  
int nr;
- Sonstiges:** Der Menüoberpunkt wird anschließend punktiert dargestellt.
- Referenz:** Siehe auch OnMenu

## 7.1.6 OnMenu

|            |                                                             |                                                     |
|------------|-------------------------------------------------------------|-----------------------------------------------------|
| Syntax:    | OnMenu(windowptr,nr);                                       |                                                     |
| Funktion:  | Schaltet einen Menüoberpunkt ein.                           |                                                     |
| Parameter: | windowptr                                                   | -> Zeiger auf das Window, zu dem das Menü gehört.   |
|            | nr                                                          | -> Nummer des Menüs, das eingeschaltet werden soll. |
| Ergebnis:  | Kein Ergebnis.                                              |                                                     |
| Datentyp:  | struct Window *windowptr;<br>int nr;                        |                                                     |
| Sonstiges: | Der Menüoberpunkt wird nun nicht mehr punktiert dargestellt |                                                     |
| Referenz:  | Siehe auch OffMenu                                          |                                                     |

## 7.1.7 SetMenuStrip

|            |                                                                                                                                        |                                                                    |
|------------|----------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|
| Syntax:    | SetMenuStrip(windowptr,menu);                                                                                                          |                                                                    |
| Funktion:  | Setzt die Menüs, die zu dem spezifizierten Window gehören.                                                                             |                                                                    |
| Parameter: | windowptr                                                                                                                              | -> Zeiger auf das Window, für das die Menüs gesetzt werden sollen. |
|            | menu                                                                                                                                   | -> Zeiger auf die Menu-Structure des ersten Menüs der Liste.       |
| Ergebnis:  | Kein Ergebnis.                                                                                                                         |                                                                    |
| Datentyp:  | struct Window *windowptr;<br>struct Menu *menu;                                                                                        |                                                                    |
| Sonstiges: | Um mehrere Menüoberpunkte zu bekommen, muß in der Menu-Structure des ersten Menüs ein Zeiger auf das zweite Menu und so weiter stehen. |                                                                    |
| Referenz:  | Siehe auch ClearMenuStrip                                                                                                              |                                                                    |

## 7.1.8 SUBNUM

|            |                                                                                            |
|------------|--------------------------------------------------------------------------------------------|
| Syntax:    | <code>nr = SUBNUM(code);</code>                                                            |
| Funktion:  | Ermittelt die Nummer des Subitems, das angewählt wurde.                                    |
| Parameter: | <code>code</code> -> Code-Wert aus der Message-Structure der Message, die übergeben wurde. |
| Ergebnis:  | <code>nr</code> -> Nummer des Subitems, das angewählt wurde.                               |
| Datentyp:  | SHORT code;<br>int nr;                                                                     |
| Sonstiges: | Die Subitems sind aufsteigend von 0 an numeriert.                                          |
| Referenz:  | Siehe auch Kapitel 7.3 Die Abfrage von Menüs, MENUNUM, ITEMNUM                             |

## 7.2 Subitems, Command-Tasten, Grafiken und MutualExclude

Soll ein Item weitere Subitems besitzen, so trägt man einfach in die MenuItem-Structure in SubItem den Zeiger auf die MenuItem-Structure des Subitems ein.

Soll einem Item eine alternative Tastensequenz zugeordnet werden, so trägt man das entsprechende Zeichen in »Command« in der MenuItem-Structure des entsprechenden Items ein. Die Zeichensequenz besteht dann aus Amiga + Zeichen. Zusätzlich muß noch das COMMSEQ-Flag gesetzt werden.

Soll ein Item nicht als Text, sondern als Grafik dargestellt werden, so wird das ITEMTEXT-Flag nicht gesetzt und in ItemFill wird der Zeiger auf die Image-Structure der Grafik eingetragen, die das Item repräsentiert. Soll dieses Item dann durch eine andere Grafik dargestellt werden, wenn der Mauszeiger über ihm steht, so ist das Flag HIGHIMAGE zu setzen und in SelectFill der Zeiger auf das alternative Image einzutragen.

In der Menu-Item-Structure gibt es einen Eintrag, der MutualExclude (gegenseitiger Ausschluß) genannt wird. An dieser Stelle kann ein Eintrag vorgenommen werden, der beschreibt, welche Items sich untereinander bedingen. Dies ist natürlich nur bei Items sinnvoll, die Attribute darstellen.

Wie MutualExclude gesetzt wird, wollen wir anhand eines Beispiels darlegen:

MutualExclude ist bitweise zu betrachten. Bit 0 bezieht sich auf das erste Item in dem Menükasten. Bit 1 auf das zweite usw.

Wir haben nun drei Items in dem Menükasten stehen. Item 2 und Item 3 dürfen gleichzeitig »eingeschaltet« sein. Ist Item 3 eingeschaltet, so darf kein weiteres eingeschaltet werden. Ebenso darf Item 3 nicht eingeschaltet werden, wenn Item 1 oder 2 eingeschaltet sind. Daraus folgt:

```
für Item 1: 000000000000100 = 0x0004
für Item 2: 00000000000100 = 0x0004
für Item 3: 11111111111011 = 0xFFFF
```

Diese Werte müssen nun in die MutualExclude-Felder der entsprechenden MenuItem-Structures eingetragen werden.

## 7.3 Die Abfrage von Menüs

Die Abfrage von Menüs geschieht über die Window-Structure des Windows, zu dem die Menüs gehören.

Wie schon von vorhergehenden Demonstrationen her bekannt, muß zuerst über das Window abgefragt werden, ob eine Menüauswahl stattgefunden hat:

```
if(message = (struct IntuiMessage *) GetMsg(windowptr->UserPort))
{
 MessageClass = message->Class; /* Messageüberrettet */
 code = message->Code; /* werden, da sie durch */
 ReplyMsg(message); /* Reply zerstört werden */
 if(MessageClass == MENUPIK) menu(); /* Sprung zur eigenen */
}; /* Menüabfrage s.u. */
```

MENUPIK ist ein IDCMP-Flag. Auf eben diese Weise kann auch jede andere Meldung abgefragt werden. Dabei ist dann natürlich nicht nach MENUPIK zu fragen, sondern nach dem entsprechenden anderen Flag.

Anschließend folgt nun die Auswertung. Diese kann bei den Menüs nach zwei verschiedenen Methoden geschehen:

1. Es wird die Menü- und die Itemnummer über MENUNUM(code) und ITEMNUM(code) ermittelt. Wenn die von MENUNUM ermittelte Nummer nicht gleich MENUNULL ist, wird anschließend die Adresse der MenuItem-Structure des angewählten Items mittels ItemAdress ermittelt. Diese Adresse ist dann mit den Adressen der eigenen MenuItem-Structures zu vergleichen, bis die richtige herausgefunden ist.

Der Nachteil dieser Methode liegt darin, daß die SubItems dadurch nicht abgefragt werden können.

2. Zum anderen kann direkt mit den Nummern gearbeitet werden, die Intuition den Menüs, Items und SubItems gibt. Die Menüs, Items und Subitems werden in der Reihenfolge numeriert, in der sie auf dem Schirm erscheinen. Jeweils mit 0 beginnend und von oben nach unten bzw. von links nach rechts.

Wenn also ein Menü mit drei Menüoberpunkten erstellt wird, wovon das erste zwei, das zweite ein und das dritte drei Item hat, sieht die Abfrage folgendermaßen aus:

Zusätzlich gehen wir noch davon aus, daß das zweite Item des dritten Menüs ein Subitem hat.

```
menu()
{
 if(MENUNUM(code) != MENUNULL)
 switch(MENUNUM(code))
 {
 case 0: switch(ITEMNUM(code))
 {
 case 0:
 case 1:
 };
 case 1: switch(ITEMNUM(code))
 {
 case 0:
 };
 case 2: switch(ITEMNUM(code))
 {
 case 0:
 case 1: switch(SUBNUM(code))
 {
 case 0:
 };
 case 2:
 };
 };
 };
}
```

Hinter den Case-Anweisungen und der entsprechenden Nummer können nun die entsprechenden Befehle oder Funktionsaufrufe stehen, die nach Betätigung des Items ausgeführt werden sollen.

```

1 /*****
2
3 Menue - Demonstration
4 last update 26/05/87
5 von Joerg Koch und Frank Kremser
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Deffnet verschieden Menues und zeigt die unterschiedliche Darstellungsart
11 der Menues
12
13 *****/
14
15 #include <exec/types.h> /* Include - Files */
16 #include <exec/memory.h>
17 #include <intuition/intuition.h>
18
19 struct IntuitionBase *IntuitionBase; /* Lib/Window-Zeiger */
20 struct GfxBase *GfxBase;
21 struct Window *w;
22
23 UWORD image[] = /* Disk / Image fuer Menue */
24 {
25 0x8001,
26 0x7FFE,
27 0x7FFE,
28 0x7C3E,
29 0x6006,
30 0x7C3E,
31 0x7FF2,
32 0x7FFE,
33 0x8001
34 };
35
36 struct Image image31 = /* Image Struktur */
37 {
38 0, /* Linke Ecke */
39 0, /* obere Ecke */
40 16, /* Breite */
41 9, /* Hoehe */
42 1, /* Tiefe */
43 &image[0], /* Zeiger auf Image-Data */
44 1, /* Plane Pick */
45 0, /* Plane OnOff */
46 NULL, /* Naechstes Image */
47 };
48
49 struct MenuItem item31 = /* MenuItem Struktur */
50 {
51 NULL, /* naechstes MenuItem */
52 -5, /* linke Ecke */
53 0, /* obere Ecke */
54 50, /* Breite */
55 11, /* Hoehe */
56 ITEMENABLED:HIGHCOMP, /* Flags */
57 0, /* MutalExclude */
58 (APTR) &image31, /* ImageData */
59 NULL, /* Command Key */
60 NULL,
61 NULL
62 };
63
64 struct Menu menu3 = /* Menu / Struktur */

```

```

65 {
66 NULL, /* naechstes Menu */
67 300, /* linke Ecke */
68 0, /* obere Ecke */
69 50, /* Breite */
70 10, /* Hoehe */
71 MENUEENABLED, /* Flags */
72 "Grafik", /* Menu-Name */
73 &item31 /* Erstes MenuItem */
74 };
75
76 struct IntuiText text21 = /* IntuiText Struktur */
77 {
78 0, /* DetailPen */
79 1, /* BlockPen */
80 JAM1, /* DrawMode */
81 0, /* Linke Ecke */
82 0, /* Obere Ecke */
83 NULL,
84 (UBYTE *) "Joerg Koch", /* Text */
85 NULL /* Zeiger auf naechsten IntuiText */
86 };
87
88 struct IntuiText text22 =
89 {
90 0,
91 1,
92 JAM1,
93 0,
94 0,
95 NULL,
96 (UBYTE *) "Frank Kremser",
97 NULL
98 };
99
100 struct MenuItem item21 =
101 {
102 NULL,
103 -5,
104 22,
105 116,
106 11,
107 CHECKIT:ITEMTEXT:ITEMENABLED:HIGHBOX,
108 0,
109 (APTR) &text21,
110 NULL,
111 NULL,
112 NULL
113 };
114
115 struct MenuItem item22 =
116 {
117 &item21,
118 -5,
119 0,
120 116,
121 11,
122 ITEMTEXT:ITEMENABLED:HIGHBOX,
123 0,
124 (APTR) &text22,
125 NULL,
126 NULL,
127 NULL
128 };

```

```
129
130 struct Menu menu2 =
131 {
132 &menu3,
133 150,
134 0,
135 120,
136 10,
137 MENUENABLED,
138 "Demo von",
139 &item22
140 };
141
142 struct IntuiText text11 =
143 {
144 0,
145 1,
146 JAM1,
147 0,
148 0,
149 NULL,
150 (UBYTE *) "Menu-Demo",
151 NULL
152 };
153
154 struct IntuiText text12 =
155 {
156 0,
157 1,
158 JAM1,
159 0,
160 0,
161 NULL,
162 (UBYTE *) "Programmende",
163 NULL
164 };
165
166 struct IntuiText subtext =
167 {
168 0,
169 1,
170 JAM1,
171 0,
172 0,
173 NULL,
174 (UBYTE *) "\ Ende Aq", /* wird die Tastenkombination Amiga + q */
175 NULL /* betaetigt, so beendet dies das Programm */
176 };
177
178 struct MenuItem subitem =
179 {
180 NULL,
181 60,
182 10,
183 116,
184 11,
185 ITEMTEXT:ITEMENABLED:HIGHCMP,
186 0,
187 (APTR) &subtext,
188 NULL,
189 113,
190 NULL,
191 NULL
192 };
```

```

193
194 struct MenuItem item12 =
195 {
196 NULL,
197 -5,
198 0,
199 116,
200 11,
201 ITEMTEXT:ITEMENABLED:HIGHCOMP,
202 0,
203 (APTR) &text12,
204 NULL,
205 NULL,
206 &subitem,
207 NULL
208 };
209
210 struct MenuItem item11 =
211 {
212 &item12,
213 -5,
214 22,
215 116,
216 11,
217 ITEMTEXT:ITEMENABLED:HIGHCOMP,
218 0,
219 (APTR) &text11,
220 NULL,
221 NULL,
222 NULL
223 };
224
225 struct Menu menu1 =
226 {
227 &menu2,
228 10,
229 0,
230 120,
231 10,
232 MENUENABLED,
233 "Programmende",
234 &item11
235 };
236
237 struct NewWindow nw = /* NewWindow / Struktur */
238 {
239 0, /* Linke Ecke */
240 0, /* Obere Ecke */
241 640, /* Breite */
242 256, /* Hoehe */
243 0, /* DetailPen */
244 1, /* BlockPen */
245 MENUPICK, /* IDCMP-Flags */
246 ACTIVATE:WINDOWDEPTH:WINDOWDRAG:NOCAREREFRESH, /* Flags */
247 NULL,
248 NULL,
249 NULL,
250 NULL,
251 NULL,
252 0,
253 0,
254 0,
255 0,
256 WBENCHSCREEN /* Screen Typ */

```

```
257);
258
259 struct IntuiMessage *message;
260 ULONG MessageClass;
261 USHORT code;
262
263 main()
264 {
265 /* Oeffnen der Libs */
266 IntuitionBase = (struct IntuitionBase *)
267 OpenLibrary("intuition.library",0L);
268 if (IntuitionBase == NULL) exit();
269
270 GfxBase = (struct GfxBase *) OpenLibrary("graphics.library" , 0L);
271 if (GfxBase == NULL) exit();
272
273 /* Oeffnen des Window */
274 w = (struct Window *) OpenWindow(&nw);
275 if(w == NULL) exit();
276
277 SetMenuStrip(w,&menu1); /* Setzen des Menues */
278
279 for(;;) /* Endlosschleife */
280 if(message = (struct IntuiMessage *)GetMsg(w->UserPort))
281 {
282 /* Message abwarten und Auswerten */
283 MessageClass = message->Class; /* Message retten */
284 code = message->Code;
285 ReplyMsg(message); /* Wenn Menu angewaehlt, dann menu() */
286 if(MessageClass == MENU PICK) menu();
287 }
288
289 menu()
290 {
291 /* Wenn regulaeres Menu angewaehlt, dann..*/
292 if(MENUNUM(code) != MENU NULL)
293 switch(MENUNUM(code)) /* Welches Menue wurde Ausgewaehlt ????? */
294 {
295 case 0: switch(ITEMNUM(code)) /* Welches Item ????? */
296 {
297 case 0: break;
298 case 1: switch(SUBNUM(code))
299 {
300 case 0: CloseWindow(w);
301 CloseLibrary(GfxBase);
302 CloseLibrary(IntuitionBase);
303 exit();
304 break;
305 }
306 }
307 break;
308 case 1: switch(ITEMNUM(code))
309 {
310 case 0: break;
311 case 1: break;
312 }
313 break;
314 case 2: switch(ITEMNUM(code))
315 {
316 case 0: break;
317 }
318 break;
319 }
320 }
```

# Die Gadgets

Das Gadget, wörtlich übersetzt Dingsda bzw. Apparat, ist ein sehr nützliches, von Intuition bereitgestelltes Hilfsmittel, welches die Bedienung von Programmen, Windows oder Screens mit der Maus unkompliziert und bedienungsfreundlich macht.

Bei Gadgets bestehen zwei Möglichkeiten, sie einzusetzen. Zum einen können sogenannte System-Gadgets in einem eigenen Programm eingesetzt werden. Solche sind z.B. die Schließ- und Zurück-Gadgets von Windows. Die zweite Möglichkeit ist die Verwendung von selbstdefinierten Gadgets. Diese bieten besondere Reize. Da der Programmierer zwischen vier verschiedenen Grundtypen auswählen kann, sind diese Gadgets besonders anpassungsfähig:

## **BOOLEAN-Gadget**

Ja- oder Nein-Entscheidungen können mit einem Boolean-Gadget abgefragt werden. Dies sind sehr einfache Gadgets, die durch Verwendung eigener Images (Grafiken) faszinierende Effekte in eigenen Programmen hervorrufen. So kann z.B. das Image ein Schalter sein, der dann bei Betätigung durch das Anklicken mit der Maus in eine andere Stellung springt und bei Abfrage »Ja = Schalter betätigt«, »Nein = Schalter nicht betätigt« beispielsweise zwischen 60 oder 80 Zeichen pro Zeile umschaltet.

## **PROPORTIONAL-Gadget**

Soll sich ein Wert mit dem Verschieben eines Gadgets verändern, so bietet sich die Verwendung von Proportional-Gadgets an. Ein Beispiel hierfür sind die allseits bekannten Schiebe-Regler für Farb-Werte.

### **TEXT-Gadgets**

Gadgets bieten auch die Möglichkeit der Eingabe von Texten. Hier wird zwischen String-Gadgets und Integer-Gadgets unterschieden. Mit String-Gadgets kann der Anwender verschiedene Texte übergeben, wobei ihm eine begrenzte Anzahl von Editier-Funktionen zur Verfügung stehen. Integer-Gadgets sind eine Abwandlung der String-Gadgets. Bei Integer-Gadgets können aber nur, wie der Name schon sagt, Integer-Werte übergeben werden.

### **SYSTEM-Gadgets**

Zu diesen selbst definierbaren Gadgets kommen noch die schon erwähnten System-Gadgets. Diese werden automatisch von Intuition an den jeweiligen Screen oder je nach Wunsch an die jeweilige Window »geheftet«. Sie haben eine feste Position und können in ihrer Funktion nicht verändert werden. Auch ihr Aussehen ist immer gleich. Nähere Informationen zu diesen Gadgets finden Sie im Kapitel Window-Gadgets.

## 8.1 Die Gadget-Structure

Um mit den Gadgets arbeiten zu können, muß zunächst eine allgemeine Structure, die für alle Gadget-Typen gleich ist, an Intuition übergeben werden:

```
struct Gadget
{
 struct Gadget *NextGadget;
 SHORT LeftEdge, TopEdge, Width, Height;
 USHORT Flags;
 USHORT Activation;
 USHORT GadgetType;
 APTR GadgetRender;
 APTR SelectRender;
 struct IntuiText *GadgetText;
 LONG MutualExclude;
 APTR SpecialInfo;
 USHORT GadgetID;
 APTR User;
}
```

**NextGadget:**

Hier muß der Pointer auf die nächste Gadget-Structure eingetragen werden. Wenn dies aber das letzte Gadget in dieser Liste ist, so muß diese Variable auf »NULL« gesetzt werden.

**LeftEdge, TopEdge  
Width, Height:**

Diese vier Variablen legen die Position und Dimension des jeweiligen Gadgets fest, die es in einem Requester oder Window annehmen soll. Durch Setzen eines bestimmten Flags in der Variable »Flags« können die Werte jeweils vom linken, oberen Rand oder aber vom rechten, unteren Rand aus angegeben werden.

LeftEdge gibt die horizontale Position des Gadgets in Bezug auf das jeweilige Window oder Requester an.

Wird die Variable GRELRIGHT in Flags gesetzt, so muß LeftEdge einen negativen Wert von z.B. -10 annehmen. Das Gadget wird dann 10 Pixels von dem augenblicklichen rechten Rand des Requesters oder Windows gezeichnet. Ist GRELRIGHT nicht gesetzt, so muß LeftEdge positiv sein. Dann wird das Gadget vom linken Rand aus gezeichnet.

TopEdge gibt die vertikale Position des Gadgets an.

Wenn das Gadget vom unteren Ende des z.B. Windows gezeichnet werden soll, muß in Flags das Flag GRELBOTTOM gesetzt sein. TopEdge muß dazu natürlich negativ angegeben werden. Wird GRELBOTTOM nicht gesetzt, so muß TopEdge positiv sein. Die Position des Gadgets richtet sich dann nach dem oberen Rand des Windows oder Screens.

Width und Height geben die Breite und Höhe des Gadgets an. Ob die Werte positiv oder negativ sein müssen, wird in Flags durch Setzen der Flags GRELWIDTH und GRELHEIGHT angegeben. Ist eines der Flags gesetzt, so muß der entsprechende Wert negativ angegeben werden.

Sind die Werte negativ und die Flags sind gesetzt, verändert sich die Größe des Gadgets in Bezug auf die Größe des Windows. Sind die Werte positiv und die Flags nicht gesetzt, bleibt das Gadget in seiner Größe unverändert, auch wenn das Window verändert wird.

Flags:

Durch das Setzen von verschiedenen Flags können die Eigenschaften von Gadgets festgelegt werden :

Was beim Anklicken des Gadgets passieren soll, wird durch zwei GADGHIGHBITS-Flags festgelegt. Es bestehen vier verschiedene Kombinationsmöglichkeiten, von denen nur eine verwendet werden darf:

**GADGHCOMP:** Dieses Flag sagt aus, daß beim Anklicken des Gadgets alle nicht dargestellten Bits gezeigt werden.

**GADGHBOX:** Wird das Gadget angeklickt, so wird eine Box um das Gadget gezeichnet.

**GADGHIMAGE:** Wenn das Gadget angeklickt wird, so weist dieses Flag darauf hin, das nun ein neues Image oder Border gezeichnet werden soll.

**GADGHNONE:** Dieses Flag muß gesetzt sein, wenn kein Verändern des Gadgets beim Anklicken durch die Maus erwünscht wird.

Folgende Flags bestimmen das Aussehen, sowie das Verhalten der Größe der Gadgets :

**GADGIMAGE:** Flag muß gesetzt sein, wenn GadgetRender nicht auf »NULL« gesetzt wird.

**GRELBOTTOM,  
GRELRIGHT,  
GRELWIDTH,  
GRELHEIGHT:** Diese vier Flags geben die Größe und Position des Gadgets im Bezug z.B. auf ein Window wieder. Siehe TopEdge, LeftEdge, Width und Height.

**SELECTED:** Mit diesem Flag kann der Anfangszustand des Activation-Flags TOGGLESELCT bestimmt werden. Ist SELECTED gesetzt, so wird das Gadget im angeklickten Zustand dargestellt.

**GADGDISABLED:** Wenn dieses Flag gesetzt ist, wird das Gadget nicht dargestellt. Der Zustand dieses Flags kann mit den Befehlen `OffGadget` und `OnGadget` geändert werden.

**Activation:** Hier können verschiedene Flags gesetzt werden, die die Benutzung und Aktivierung der Gadgets beschreiben :

**TOGGLESELECT:** Ist dieses Flag gesetzt, so wechselt das Aussehen des Gadgets mit dem Anklicken der Maus. Der Anfangszustand des Gadgets kann durch das Flag `SELECTED` in `Flags` bestimmt werden.

**GADGIMMEDIATE:** Ist dieses Flag gesetzt, wird dem Programm ständig mitgeteilt, ob das Gadget angeklickt ist. Dabei muß nicht unbedingt der Mauszeiger über dem Gadget sein. Nur das Setzen dieses Flags reicht nicht aus, um 100 prozentig sicher zu sein, daß der Benutzer das Gadget betätigt hat.

Ein zusätzliches Setzen von `RELVERIFY` schafft da Sicherheit.

- RELVERIFY:** Dem Programm wird das Anklicken des Gadgets nur dann mitgeteilt, wenn sich dabei der Mauszeiger über dem Gadget befindet.
- ENDGADGET:** Dieses Flag wird nur bei Requesters benutzt. Wird ein Gadget mit diesem Flag betätigt, so verschwindet das jeweilige Requester.
- FOLLOWMOUSE:** Wird ein Gadget mit diesem Flag betätigt, so erhält das Programm die Position der Maus und das Gadget wird mit dem Mauszeiger verschoben.

Diese vier Flags werden in Bezug auf Windows eingesetzt :

- RIGHTBORDER:** Das Gadget wird in die rechte Umrandung eingesetzt.
- LEFTBORDER:** Das Gadget wird in die linke Umrandung eingesetzt.
- BOTTOMBORDER:** Das Gadget wird in die untere Umrandung eingesetzt.
- TOPBORDER:** Das Gadget wird in die obere Umrandung eingesetzt.

Die nächsten vier Flags sind wichtig bei der Verwendung von String-Gadgets:

- STRINGRIGHT:** Wenn das String-Gadget angeklickt ist, wird der Text rechtsbündig dargestellt.

|                      |                                                             |                                                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | <b>STRINGCENTER:</b>                                        | Wenn das String-Gadget angeklickt ist, wird der Text mittig dargestellt.                                                                                                                                                        |
|                      | <b>LONGINT:</b>                                             | Ist dieses Flag gesetzt, kann der Anwender in dem jeweiligen String-Gadget 32-Bit Integer-Zahlen eingeben.                                                                                                                      |
|                      | <b>ALTKEYMAP:</b>                                           | Dieses Flag muß gesetzt sein, wenn ein eigenes Key-Map in der StringInfo-Structure verwendet wird.                                                                                                                              |
| <b>GadgetType:</b>   | <b>GadgetType</b> gibt den Typ des verwendeten Gadgets an : |                                                                                                                                                                                                                                 |
|                      | <b>BOOLGADGET:</b>                                          | Dieses Flag setzt ein Wahr- oder Falsch-Gadget.                                                                                                                                                                                 |
|                      | <b>STRGADGET:</b>                                           | Dieses Flag setzt ein String/Integer-Gadget.                                                                                                                                                                                    |
|                      | <b>PROPGADGET:</b>                                          | Dieses Flag setzt ein Proportional-Gadget.                                                                                                                                                                                      |
|                      | <b>GZZGADGET:</b>                                           | Dieses Flag setzt ein Gadget in einem Gimmezerozero-Window.                                                                                                                                                                     |
|                      | <b>REQGADGET:</b>                                           | Dieses Flag setzt ein Gadget, das in einem Requester verwendet werden soll.                                                                                                                                                     |
| <b>GadgetRender:</b> |                                                             | Wenn <b>GADGHIMAGE</b> bei Flags gesetzt ist, muß hier der Pointer auf das jeweilige Image oder Border eingetragen werden. Andernfalls ist die Variable »NULL«. Dieses Image wird gezeigt, wenn das Gadget nicht aktiviert ist. |

- SelectRender:** Wenn GADGHIMAGE bei Flags gesetzt ist, muß hier der Pointer auf das jeweilige Image oder Border eingetragen werden. Andernfalls ist die Variable »NULL«. Dieses Image wird gezeigt, wenn das Gadget aktiviert ist.
- GadgetText:** Wenn Text bei der Darstellung des Gadgets verwendet werden soll, muß hier der Pointer auf die jeweilige IntuitionText-Structure eingetragen werden. Die Koordinaten des Textes beziehen sich auf die Größe des Gadgets.
- MutualExclude:** Hier kann beschrieben werden, welche Gadgets sich untereinander bedingen.
- SpecialInfo:** Wird ein Proportional- oder Text/Integer-Gadget verwendet, muß hier der Pointer auf die PropInfo- bzw. StringInfo-Structure eingetragen werden. Andernfalls ist diese Variable »NULL«.
- GadgetID:** Hier wird die Gadget-Identität eingetragen, die später abgefragt und somit herausgefunden werden kann, welches Gadget betätigt wurde. Die Gadget-Identität besteht aus einer Nummer.
- User:** Hier kann ein Pointer auf eigene spezielle Daten eingetragen werden.

## 8.2 Das Boolean-Gadget

Das Boolean-Gadget ist eines der simpelsten Gadgets, die Intuition zur Verfügung stellt. Mit ihm können zwei Zustände abgefragt werden: »Wahr«, das Gadget wurde angeklickt und »Falsch«, das Gadget wurde nicht angeklickt.

Bei der Verwendung von Boolean-Gadgets wird nur die Gadget-Structure benötigt. Die Verwendung von Boolean-Gadgets wird in der Gadget-Structure in der Variable »Gadgettype« durch Setzen des BOOLGADGET-Flag festgelegt.

Wenn der Benutzer das Boolean-Gadget mit der Maus angeklickt hat, bestehen verschiedene Möglichkeiten, diesen aktivierten Zustand darzustellen:

Durch Setzen des Flags GADGHBOX in der Variable Flags der Gadget-Structure wird eine einfache Border um die Gadget-Box gezeichnet.

Wenn das Flag GADGHCOMP in der Variable Flags gesetzt wird, werden alle Bits des Gadgets komplementiert, wenn es aktiviert wurde.

Das Flag GADGHIMAGE weist darauf hin, daß bei Aktivierung des Gadgets ein neues Image gezeichnet werden soll.

Wird GADGHNONE in der Variable Flags eingetragen, findet keine Veränderung statt.

Wird in der Variable »Activation« der Gadget-Structure das TOGGLESELECT-Flag nicht gesetzt, so werden die obengenannten Veränderungen nur kurz, d.h. so lange wie der Benutzer die Maustaste beim Selektieren des Gadgets gedrückt hat, angezeigt. Durch Setzen dieses Flags wechselt der Zustand des Gadgets nach jedem Anklicken mit der Maustaste.

## 8.3 Das Text/Integer-Gadget

Für die Benutzung von Text- bzw. Integer-Gadgets muß die bisher kennengelernte Gadget-Structure durch eine zusätzliche Structure erweitert werden, die StringInfo-Structure. Der Pointer auf diese Structure muß in der Gadget-Structure bei SpecialInfo eingetragen werden. Zudem muß in GadgetType STRGADGET stehen. Möchte man ein Integer-Gadget verwenden, so wird dies durch das zusätzliche Setzen des Flags LONGINT in der Variable Activation der Gadget-Structure festgelegt.

```
struct StringInfo
{
 UBYTE *Buffer;
 UBYTE *UndoBuffer;
 SHORT BufferPos;
 SHORT MaxChars;
 SHORT DispPos;
 SHORT UndoPos;
 SHORT NumChars;
 SHORT DispCount;
 SHORT CLeft, CTop;
 struct Layer *LayerPtr;
 LONG LongInt;
 struct KeyMap *AltKeyMap;
};
```

- Buffer:** Hier muß ein Pointer zu einem Puffer eingetragen werden, der den jeweiligen String aufnehmen soll. Der muß 0-terminated sein, also mit 0 enden.
- UndoBuffer:** Hier muß ein Pointer zu einem Puffer eingetragen werden, der so lang ist, wie der Puffer für den jeweiligen String. Der UndoBuffer ermöglicht, die augenblickliche Eingabe rückgängig zu machen.
- BufferPos:** Diese Variable gibt die darzustellende Position des Cursors an, wenn der jeweilige String initialisiert ist.
- MaxChars:** Hier muß die maximale Anzahl der Buchstaben/Zahlen eingetragen werden, inklusive der 0 am Ende, da der String »0-terminated« sein muß.
- DispPos:** DispPos gibt die Buffer-Position der ersten darzustellenden Zahl bzw. Buchstaben an.
- Diese Variablen werden von Intuition initialisiert :
- UndoPos:** Gibt die Buchstaben/Zahlen-Position in dem UndoBuffer an.

|              |                                                                                                                                                                                                                                |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NumChars:    | NumChars enthält die Anzahl der Buchstaben/Zahlen im Puffer.                                                                                                                                                                   |
| DispCount:   | Dies enthält die Anzahl der sichtbaren Buchstaben/Zahlen im String-Gadget, die auf dem Bildschirm dargestellt werden.                                                                                                          |
| CLeft, CTop: | Diese beiden Variablen enthalten die Position der Box in dem das String-Gadget enthalten ist.                                                                                                                                  |
| LayerPtr:    | Dies gibt den Layer (»Grafik-Schicht«) an, die das Gadget enthält.                                                                                                                                                             |
| LongInt:     | Wenn dies ein Integer-Gadget ist, enthält diese Variable die Integer-Zahl.                                                                                                                                                     |
| AltKeyMap:   | Wenn kein eigenes KeyMap verwendet wird, muß hier »NULL« eingetragen werden. Andernfalls wird hier der Pointer auf das eigene KeyMap eingetragen. Dazu muß das Activation-Flag ALTKEYMAP in der Gadget-Structure gesetzt sein. |

String-Gadgets erlauben dem Benutzer die Eingabe von Texten oder Zahlen. Es können mehrere Text- und Integer-Gadgets dargestellt werden. Aktiv kann nur jeweils eins sein.

String-Gadgets können zwei Puffer enthalten. Einen Puffer für den einzugebenden String und einen für den zuletzt eingegebenen String, falls die Eingabe rückgängig gemacht werden soll.

Wenn ein String-Gadget einen »UndoBuffer« enthält, so wird beim Anklicken des Text/Integer-Gadgets der dargestellte String in diesen Puffer kopiert. Verschiedene Tasten-Kombinationen bieten eine komfortable Editierung des Textes bzw. der Zahl :

Rechte Amiga-Taste  
und Q:

Macht die letzte Eingabe rückgängig.

Rechte Amiga-Taste  
und X:

Löscht den Eingabe-Puffer.

**Return**:

Beendet die Eingabe.

**Backspace**:

Löscht die Buchstaben links vom Cursor.

**Del**:

Löscht die Buchstaben unter dem Cursor.

**Shift** **←** oder **→**:

Bewegt den Cursor zum Anfang oder Ende des Gadgets.

☐← oder ☐→: Bewegt den Cursor im Text nach links oder rechts.

Durch zwei Flags kann der String, der auf dem Bildschirm dargestellt wird, justiert werden:

Das Flag `STRINGCENTER` in der Variable `Activation` in der Gadget-Structure gibt an, daß der Text mittig dargestellt werden soll.

Das Flag `STRINGRIGHT` justiert den Text rechtsbündig.

## 8.4 Das Proportional-Gadget

Wenn Proportional-Gadgets verwendet werden sollen, muß die Gadget-Structure um eine weitere Struktur, die PropInfo-Structure, erweitert werden. Der Pointer dieser Structure, wird dann in die Variable SpecialInfo der Gadget-Structure eingetragen. Zudem muß in GadgetType PROPGADGET eingesetzt werden.

```
struct PropInfo
{
 USHORT Flags;
 USHORT HorizPot;
 USHORT VertPot;
 USHORT HorizBody;
 USHORT VertBody;
 USHORT CWidth;
 USHORT CHeight;
 USHORT HPotRes, VPotRes;
 USHORT LeftBorder;
 USHORT TopBorder;
};
```

|                 |                                                                                                                   |
|-----------------|-------------------------------------------------------------------------------------------------------------------|
| Flags:          | Durch die Eingabe von verschiedenen Flags kann das Aussehen und die Bewegung des Prop-Gadgets festgelegt werden : |
| FREEHORIZ:      | Erlaubt die Bewegung in die horizontale Richtung.                                                                 |
| FREEVERT:       | Erlaubt die Bewegung in die vertikale Richtung.                                                                   |
| AUTOKNOB:       | Die Verwendung eines Auto-Knopfes.                                                                                |
| KNOBHIT:        | Dies wird von Intuition gesetzt, wenn der Knopf angeklickt wird.                                                  |
| PROPBORDERLESS: | Wenn dieses Flag gesetzt ist, wird keine Umrandung um das Gadget gezeichnet.                                      |
| HorizPot:       | Gibt die horizontale Position des Knopfes beim Initialisieren des Prop-HGadgets an.                               |
| VertPot:        | Gibt die vertikale Position des Knopfes beim Initialisieren des Prop-Gadgets an.                                  |

|            |                                                                       |
|------------|-----------------------------------------------------------------------|
| HorizBody: | Gibt die horizontale prozentuale Schrittweite in Bezug auf 0xFFFF an. |
| VertBody:  | Gibt die vertikale prozentuale Schrittweite im Bezug auf 0xFFFF an.   |

Diese Werte werden von Intuition gesetzt:

|                   |                                                      |
|-------------------|------------------------------------------------------|
| CWidth:           | Breite des Raums in dem sich das Prop-Gadget bewegt. |
| CHeight:          | Höhe des Raums in dem sich das Prop-Gadget bewegt.   |
| HPotRes, VPotRes: | Erhöhung der vertikalen und horizontalen Werte.      |
| LeftBorder:       | Linker Rand des Proportional-Gadgets.                |
| TopBorder:        | Oberer Rand des Proportional-Gadgets.                |

Proportional-Gadgets sind die flexibelsten Gadgets, die Intuition zur Verfügung stellt. Man kann mit Ihnen nicht nur Verhältnisse darstellen, sondern auch Gegenstände per Maus über den Bildschirm wandern lassen.

Der Gegenstand, auch Knopf (englisch Knob) genannt, kann beliebige Formen durch die Definition eines eigenen Images annehmen. Durch Setzen des AUTOKNOB-Flags kann man aber auch einen vordefinierten Auto-Knopf verwenden. Er kann, durch das Setzen oder Nicht-Setzen der Flags FREEHORIZ und FREEVERT in der PropInfo-Struktur, in alle Richtungen bewegt oder wie ein Farbreger horizontal bzw. vertikal verschoben werden.

Sehr wichtig bei einem Proportional-Gadget ist die Angabe der Schrittweite. Die Schrittweite steht immer in einem gewissen Verhältnis zu der maximal und minimal möglichen Verschiebung auf dem Screen. Die kleinste Schrittweite des Knopfes wird in den Variablen VertBody und HorizBody der PropInfo-Structure eingegeben. Ein einfaches Beispiel für die Schrittweite bietet die Darstellung eines Farb-Reglers. Der Amiga kann maximal pro Farbwert 16 verschiedene Abstufungen annehmen. Das heißt, daß unser Gadget maximal 16 verschiedene Schritte haben darf, um eine optimale Farbabstufung darstellen zu können. HorizBody und VertBody kann maximal einen Wert von 65536 annehmen (Hex = 0xFFFF). Die kleinste Schrittweite ist bei 16 Abstufungen dann  $65536/16*1 = 4096$  (0x1000). HorizBody muß somit, wenn es sich um einen horizontalen Farb-Regler handelt, den Wert 0x1000 annehmen.

Wenn man beispielsweise einen Textpuffer, der 30 Zeilen enthält, darstellen will, auf dem Bildschirm aber nur 20 Zeilen dargestellt werden können, so sieht die Berechnung folgendermaßen aus:  $65536/30*20 = 109,2$

## 8.5 Die Abfrage von Gadgets

Bei der Abfrage, welches Gadget nun betätigt wurde, bedient man sich der Hilfe von `IntuiMessage`, das die aktuellen Daten der Ein- und Ausgabe der »Schnittstelle« (IDCMP) übergibt.

Wichtig bei unseren Gadgets ist, daß in `GadgetID` jeweils eine unterschiedliche Nummer steht, an der wir unser Gadget erkennen können. Wird ein Gadget nun betätigt und wir haben das Flag `RELVERIFY` in unserer Variable `Activation` der Gadget-Structure gesetzt, so erhalten wir von `IntuiMessage` das Gadget-Flag `GADGETUP`. Ist in unserer Gadget-Structure das Flag `GADGIMMEDIATE` gesetzt, so erhalten wir `GADGETDOWN`. Wenn diese Flags an der »Schnittstelle« vorhanden sind, brauchen wir nur noch `IntuiMessage` nach unserem Gadget-Pointer abzufragen. Wie wir dieses Flags von `IntuiMessage` abfragen, zeigt folgendes Beispiel:

```
struct IntuiMessage *message;

/* Endlosschleife */

for (;;)
{
 /* Nachricht über den, vom
 Window abgeleiteten Message-Port-Pointer holen */

 if (message = (struct IntuiMessage *)GetMsg(w->UserPort))
 {

 /* Parameter übernehmen und so lange warten, bis eine
 Nachricht am Port angekommen ist */

 MessageClass = message->Class;
 code = message->Code;
 ReplyMsg(message);

 /* Nachricht ist am Port angekommen, unsere Nachricht ?
 ja, dann Unterfunktion*/

 switch (MessageClass)
 {
 case GADGETDOWN :
 case GADGETUP : Gad(message);
 break;
 }
 }

 /* Wenn BOOL - Variable schluß = TRUE, dann Ausstieg aus
 der Endlosschleife. */
}
```

```
if (schluß = TRUE)
 exit();
}

/* Unterfunktion Gad(mess) */

Gad(mess)
struct IntuiMessage *mess;
{
 struct Gadget *gad;
 int gadid;

 /* Pointer zu meinem Gadget von IntuiMessage holen */

 gad = mess->IAddress;

 /* Gadget - Nummer holen */

 gadid = gad->GadgetID;

 /* Switch / Case abfrage, welches Gadget 0 oder 1, wenn 1 dann
 Ende */

 switch(gadid)
 {
 case 0:
 break;
 case 1: schluß = TRUE;
 break;
 };
}
```

Sicherlich könnte man auch die Unterfunktion Gad() in der Endlosschleife, wo die Flags GADGETDOWN und GADGETUP abgefragt werden, mit unterbringen. Man muß jedoch dabei bedenken, daß dadurch das ganze Geschehen sehr unübersichtlich würde.

## 8.6 Die Gadget-Befehle

### 8.6.1 AddGadget

|            |                                                                                                                                                                                                                                                                                                                                                                          |                                                                           |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|
| Syntax:    | <code>pos = AddGadget(Window, Gadget, Position);</code>                                                                                                                                                                                                                                                                                                                  |                                                                           |
| Funktion:  | Hängt das jeweilige Gadget an die Gadget-Liste des jeweiligen Window.                                                                                                                                                                                                                                                                                                    |                                                                           |
| Parameter: | Window                                                                                                                                                                                                                                                                                                                                                                   | -> Zeiger auf die Window-Structure der Window, zu der das Gadget gehört.  |
|            | Gadget                                                                                                                                                                                                                                                                                                                                                                   | -> Zeiger auf die Gadget-Structure des Gadgets, das angefügt werden soll. |
|            | Position                                                                                                                                                                                                                                                                                                                                                                 | -> Position, an der das Gadget eingefügt werden soll.                     |
| Ergebnis:  | pos                                                                                                                                                                                                                                                                                                                                                                      | -> Position, an der das Gadget eingefügt worden ist.                      |
| Datentyp:  | <code>struct Window *Window;</code><br><code>struct Gadget *Gadget;</code><br><code>int Position;</code><br><code>int pos;</code>                                                                                                                                                                                                                                        |                                                                           |
| Sonstiges: | Soll das Gadget am Ende der Gadget-Liste stehen, so empfiehlt es sich, die Position auf -1 zu setzen. Werden Systemgadgets verwendet, sind diese an erster Stelle in die Liste einzutragen. Zu beachten ist noch, daß AddGadget nur das jeweilige Gadget in die Liste einträgt und nicht auf dem Screen darstellt. Dafür muß der Befehl RefreshGadgets verwendet werden. |                                                                           |
| Referenz:  | Siehe auch RefreshGadget                                                                                                                                                                                                                                                                                                                                                 |                                                                           |

### 8.6.2 ModifyProp

|            |                                                                                                         |                                                                                         |
|------------|---------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| Syntax:    | <code>ModifyProp(PropGadget, Pointer, Requester, Flags, HorizPot, VertPot, HorizBody, VertBody);</code> |                                                                                         |
| Funktion:  | Diese Routine verändert die Parameter eines Proportional-Gadgets.                                       |                                                                                         |
| Parameter: | PropGadget                                                                                              | -> Zeiger auf die Gadget-Structure des Proportional-Gadgets, das verändert werden soll. |

- Pointer** -> Zeiger auf die Window-Structure, in dem sich das Prop-Gadget befindet.
- Requester** -> Zeiger auf die Requester-Structure, in dem sich das Prop-Gadget befindet, falls es zu einem Requester gehört. Gehört das Gadget nicht in ein Requester, kann dieser Parameter auf NULL gesetzt werden
- Flags** -> Neue Flags des Gadgets:
- AUTOKNOB =** Verwendung des Standard-Knopfes.
  - FREEHORIZ =** Knopf, der horizontal bewegt werden kann.
  - FREEVERT =** Knopf, der vertikal bewegt werden kann.
  - KNOBHIT =** Dies wird von Intuition gesetzt, falls der Knopf von der Maus angeklickt wird.
  - PROPBORDER = LESS** Dies wird gesetzt, falls keine Umrandung »Border« für das Gadget vorhanden ist.
- HorizPot** -> gibt die horizontale Schrittweite des Knopfes in Prozent wieder. Beispiel :
- In 16 Schritten soll sich der Knopf bewegen, bei Variable mit 16 Bit Länge, dies entspricht gleich einer max. Zahlenlänge von 65535, berechnet sich der Wert wie folgt:
- $$65535 / 16 = 4096$$
- = > HorizProp = 4096 oder 0x1000
- VertPot** -> wie HorizPot, nur vertikal.
- HorizBody** -> horizontale Position des Knopfes nach der Installation.

VertBody -> vertikale Position des Knopfes nach der Installation.

- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct Gadget \*PropGadget;  
struct Window \*Pointer;  
struct Requester \*Requester;  
WORD Flags, HorizPot, VertPot;  
WORD HorizBody, VertBody;
- Sonstiges:** Anschließend muß noch der Befehl RefreshGadgets gegeben werden, um das geänderte Gadget darzustellen.  
  
Zur näheren Beschreibung der Proportional-Gadgets siehe Kapitel 8.
- Referenz:** Siehe auch RefreshGadget.

### 8.6.3 OffGadget

- Syntax:** OffGadget(Gadget, Pointer, Requester);
- Funktion:** Diese Funktion schaltet das angegebene Gadget aus der spezifizierten Window-Structure, bzw. aus der Requester-Structure ab.
- Parameter:** Gadget -> Zeiger auf die Gadget-Structure des Gadgets, das abgeschaltet werden soll.  
  
Pointer -> Zeiger auf die Window-Structure, in der das Gadget eingetragen ist.  
  
Requester -> Zeiger auf die Requester-Structure, in der das Gadget eingetragen ist.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct Gadget \*Gadget;  
struct Window \*Pointer;  
struct Requester \*Requester;
- Sonstiges:** Das durch OffGadget abgeschaltete Gadget kann durch OnGadget wieder eingeschaltet werden.  
  
Wird ein Gadget durch OffGadget gelöscht, so wird es nicht aus der Gadget-Liste ausgelöscht.

Befindet sich das Gadget in einem Window, muß »Requester« auf NULL gesetzt werden. Ist das Gadget ein Requester-Gadget, so muß »Pointer« gleich NULL sein.

Referenz: Siehe auch OnGadget.

## 8.6.4 OnGadget

Syntax: OnGadget(Gadget, Pointer, Requester);

Funktion: Schaltet ein, durch OffGadget abgeschaltetes, Gadget wieder ein.

Parameter: Gadget -> Zeiger auf die Gadget-Structure des Gadgets, das eingeschaltet werden soll.

Pointer -> Zeiger auf die Window-Structure, in der das Gadget eingetragen ist.

Requester -> Zeiger auf die Requester-Structure, in der das Gadget eingetragen ist.

Ergebnis: Kein Ergebnis.

Datentyp: struct Gadget \*Gadget;  
struct Window \*Pointer;  
struct Requester \*Requester;

Sonstiges: Das mit OnGadget eingeschaltete Gadget kann mit OffGadget wieder abgeschaltet werden.

Referenz: Siehe auch OffGadget.

## 8.6.5 RefreshGadgets

Syntax: RefreshGadget(Gadgets, Pointer, Requester);

Funktion: Diese Funktion zeichnet und frischt alle Gadgets in der Gadgetliste auf, beginnend bei dem angegebenen Gadget.

Parameter: Gadget -> Zeiger auf das erste Gadget, das »refreshed« werden soll.

Pointer -> Zeiger auf die Window-Structure des Windows, in dem sich die Gadgets befinden.

Requester -> Zeiger auf die Requester-Structure des Requesters, in dem sich die Gadgets befinden.

- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct Gadget \*Gadget;  
struct Window \*Pointer;  
struct Requester \*Requester;
- Sonstiges:** Bevor RefreshGadget aufgerufen wird, müssen die jeweiligen Gadgets mit AddGadget in die GadgetListe eingetragen werden, sonst werden sie nachdem RefreshGadget aufgerufen wurde, nicht auf dem Screen dargestellt.
- Sollen alle Gadgets eines Windows, bzw. eines Requesters »refreshed« werden, so muß für Gadget folgender Ausdruck angegeben werden:
- WindowPtr -> FirstGadget
- Referenz:** Siehe auch OffGadgets

### 8.6.6 RemoveGadget

- Syntax:** Pos = RemoveGadget(Pointer, Gadget);
- Funktion:** Löschen eines Gadgets aus der jeweiligen Window-Gadget-Liste.
- Parameter:** Pointer -> Zeiger auf die Window-Structure des Windows, aus dem das Gadget gelöscht werden soll.
- Gadget -> Zeiger auf die Gadget-Structure des Gadgets, das gelöscht werden soll.
- Ergebnis:** Pos -> Position, die das Gadget in der Liste hatte.
- Datentyp:** struct Window \*Pointer;  
struct Gadget \*Gadget;  
int Pos;
- Sonstiges:** Das Image des Gadgets wird durch diese Funktion nicht vom Schirm gelöscht.
- Referenz:** Siehe auch AddGadget

```

1 /*****
2
3 Gadget-Demonstration
4 last update 26/05/87
5 von Frank Kremser und Joerg Koch
6 (c) Markt und Technik 1987
7
8 *****/
9
10 Diese Demonstration zeigt eine Auswahl an verschiedenen Gadgettypen. Mit den
11 drei Proportionalgadgets kann die Hintergrundfarbe veraendert werden.
12
13 *****/
14
15 #include <exec/types.h> /* Include-Files einlesen */
16 #include <exec/nodes.h>
17 #include <exec/lists.h>
18 #include <exec/ports.h>
19 #include <exec/devices.h>
20 #include <devices/keymap.h>
21 #include <graphics/regions.h>
22 #include <graphics/copper.h>
23 #include <graphics/gels.h>
24 #include <graphics/gfxbase.h>
25 #include <graphics/gfx.h>
26 #include <graphics/clip.h>
27 #include <graphics/view.h>
28 #include <graphics/rastport.h>
29 #include <graphics/layers.h>
30 #include <intuition/intuition.h>
31 #include <hardware/blit.h>
32
33 struct IntuitionBase *IntuitionBase; /* Lib und ander Zeiger */
34 struct GfxBase *GfxBase;
35 struct IntuiMessage *message;
36 struct RastPort *rtp;
37 struct Screen *screen;
38 struct Window *w;
39
40 /* Image des benutzerdefinierbaren Gadgets */
41
42 UWORD custimage[] =
43 {
44 0x03C0, 0x1DFB, 0x3AFC, 0x6BFE, 0x65FE, 0x5BFE, 0xEBFF, 0xFFFF,
45 0xFFFF, 0xFFFF, 0x7FFE, 0x7FFE, 0x7FFE, 0x3FFC, 0x1FFB, 0x03C0
46 };
47
48 struct Image cus_image = /* Image Structure */
49 {
50 0, /* Linke Ecke */
51 0, /* Obere Ecke */
52 16, /* Breite */
53 16, /* Hoehe */
54 1, /* Tiefe */
55 &custimage[0], /* Zeiger auf das Image */
56 1, /* PlanePick */
57 0, /* Planeonoff */
58 NULL /* Zeiger auf weitere Images */
59 };
60
61
62 struct IntuiText rtxt = /* Text zu einem Gadget */
63 {
64 1, /* DetailPen */

```

```
65 1, /* BlockPen */
66 JAM1, /* Draw-Mode */
67 -35, /* Linke Ecke */
68 2, /* Obere Ecke */
69 0, /* Zeiger auf Zeichensatz */
70 "Rot", /* Text */
71 0 /* Zeiger auf weitere Intuition-Texte */
72 };
73
74 struct IntuiText gtxt =
75 {
76 1,
77 1,
78 JAM1,
79 -35,
80 2,
81 0,
82 "Gr[n]",
83 0
84 };
85
86 struct IntuiText btxt =
87 {
88 1,
89 1,
90 JAM1,
91 -35,
92 2,
93 0,
94 "Blau",
95 0
96 };
97
98 struct IntuiText ntxt =
99 {
100 1,
101 1,
102 JAM1,
103 10,
104 -11,
105 0,
106 "Texteingabe",
107 0
108 };
109
110 struct IntuiText cus_text =
111 {
112 1,
113 1,
114 JAM1,
115 2,
116 -11,
117 0,
118 "Benutzerdefiniertes-Gadget",
119 0
120 };
121
122 struct IntuiText bool_str =
123 {
124 1,
125 1,
126 JAM1,
127 9,
128 1,
```

```

129 0,
130 "ENDE",
131 0
132 };
133
134 struct IntuiText bool2_str =
135 {
136 1,
137 1,
138 JAM1,
139 9,
140 1,
141 0,
142 "BEEP",
143 0
144 };
145
146 struct Image r_img, g_img, b_img;
147 struct PropInfo r_prop, g_prop, b_prop;
148
149 struct PropInfo cust_prop = /* Informationen zu dem Benutzer-Gadget */
150 {
151 FREEHORIZ:FREEVERT:PROPBORDERLESS, /* frei beweglich */
152 0x8000, /* Startposition X */
153 0x8000, /* Startposition Y */
154 0x800, /* Kleinst moegliche Schrittweite in X-Richtung */
155 0x800, /* Kleinst moegliche Schrittweite in Y-Richtung */
156 150, /* Reale Ausmasse des Kastens */
157 50,
158 1, /* Schrittweite in X-Richtung */
159 1, /* Schrittweite in Y-Richtung */
160 0, /* Linker Rand */
161 0 /* Rechter Rand */
162 };
163
164 UBYTE DefString[20] = "Markt & Technik";
165 UBYTE Undo [20];
166
167 struct StringInfo TexString = /* Fuer Texte in Text-Gadgets */
168 {
169 DefString, /* Zeiger auf den Puffer */
170 Undo, /* Zeiger auf den Undo-Puffer */
171 0, /* Startposition des Cursors */
172 20, /* Maximale Anzahl der Zeichen */
173 0, /* Zeiger auf den ersten Buchstaben im Puffer */
174 0, /* Position des Cursors im Undo-Puffer */
175 13, /* Anzahl der Zeichen im Puffer */
176 0, /* Anzahl der sichtbaren Zeichen */
177 0, /* Linke Ecke */
178 0, /* Obere Ecke */
179 NULL, /* Layer-Pointer */
180 0, /* Longint-Wert */
181 NULL /* Alternatives Keyboard */
182 };
183
184 USHORT Pairs[] =
185 {
186 -1,
187 -1, /* Information describing the */
188 201,
189 -1, /* border around the gadget */
190 201,
191 11,
192 -1,

```

```
193 11,
194 -1,
195 -1
196 };
197
198 USHORT Pairs1[] =
199 {
200 0,
201 0,
202 51,
203 0,
204 51,
205 11,
206 0,
207 11,
208 0,
209 0
210 };
211
212 struct Border StrBorder =
213 {
214 0, /* Linke Ecke */
215 0, /* Obere Ecke */
216 1, /* DetailPen */
217 0, /* BlockPen */
218 JAM1, /* Drawmode */
219 5, /* Anzahl der Koordinaten-Paare */
220 &Pairs1[0], /* Zeiger auf die Paare */
221 NULL /* Zeiger auf weitere Borders */
222 };
223
224 struct Border butt_border =
225 {
226 -1,
227 -1,
228 1,
229 0,
230 JAM1,
231 5,
232 &Pairs1[0],
233 NULL
234 };
235
236 struct Gadget blue_gad =
237 {
238 0, /* Zeiger auf naechstes Gadget */
239 50, /* Linke Ecke */
240 120, /* Obere Ecke */
241 200, /* Breite */
242 20, /* Hoehe */
243 GADGHCOMP, /* Flags */
244 GADGIMMEDIATE:RELVERIFY, /* Activation */
245 PROPGADGET, /* Gadget-Typ */
246 (APTR)&b_img, /* Zeiger auf Image */
247 0, /* Zeiger auf Select-Image */
248 &btxt, /* Text */
249 0, /* Mutual-Exclude */
250 (APTR)&b_prop, /* Spezial-Information: Hier PropInfo */
251 0, /* GadgetID */
252 0, /* UserData */
253 };
254
255 struct Gadget green_gad =
256 {
```

```
257 &blue_gad,
258 50,
259 100,
260 200,
261 20,
262 GADGHCOMP,
263 GADGIMMEDIATE:RELVERIFY,
264 PROPGADGET,
265 (APTR)&g_img,
266 0,
267 >xt,
268 0,
269 (APTR)&g_prop,
270 1,
271 0
272 };
273
274 struct Gadget red_gad =
275 {
276 &green_gad,
277 50,
278 80,
279 200,
280 20,
281 GADGHCOMP,
282 GADGIMMEDIATE:RELVERIFY,
283 PROPGADGET,
284 (APTR)&r_img,
285 0,
286 &rtxt,
287 0,
288 (APTR)&r_prop,
289 2,
290 0
291 };
292
293 struct Gadget tex_gad =
294 {
295 &red_gad,
296 50,
297 50,
298 200,
299 20,
300 GADGHCOMP,
301 STRINGCENTER:RELVERIFY,
302 STRGADGET,
303 (APTR)&StrBorder,
304 0,
305 &ntext,
306 0,
307 (APTR)&TexString,
308 3,
309 0
310 };
311
312 struct Gadget cust_knob =
313 {
314 &tex_gad,
315 320,
316 50,
317 300,
318 150,
319 GADGHCOMP,
320 GADGIMMEDIATE:RELVERIFY,
```

```

321 PROPGADGET,
322 (APTR)&cus_image,
323 0,
324 &cus_text,
325 0,
326 (APTR)&cust_prop,
327 4,
328 0
329 };
330
331 struct Gadget bool_gad =
332 {
333 &cust_knob,
334 51,
335 159,
336 50,
337 10,
338 GADGHCOMP,
339 GADGIMMEDIATE:RELVERIFY,
340 BOOLGADGET,
341 (APTR)&butt_border,
342 0,
343 &bool_str,
344 0,
345 0,
346 5,
347 0
348 };
349
350 struct Gadget bool2_gad =
351 {
352 &bool_gad,
353 200,
354 159,
355 50,
356 10,
357 GADGHCOMP,
358 TOGGLESELECT:GADGIMMEDIATE:RELVERIFY,
359 BOOLGADGET,
360 (APTR)&butt_border,
361 0,
362 &bool2_str,
363 0,
364 0,
365 6,
366 0
367 };
368
369 struct NewWindow nw =
370 {
371 0, /* Linke Ecke */
372 0, /* Obere Ecke */
373 640, /* Breite */
374 256, /* Hoehe */
375 0, /* DetailPen */
376 1, /* BlockPen */
377 REFRESHWINDOW:MOUSEBUTTONS:MOUSEMOVE /* IDCMP flags */
378 :GADGETDOWN:GADGETUP,
379 WINDOWDEPTH:WINDOWDRAG:REPORTMOUSE /* Flags */
380 :SMART_REFRESH,
381 &bool2_gad, /* Zeiger auf erstes Gadget des Windows */
382 NULL, /* Checkmark */
383 "Gadget-Demonstration", /* Window-Titel */
384 NULL, /* Zeiger auf screen */

```

```

385 NULL, /* Zeiger auf SuperBitMap */
386 0, /* Min. Breite */
387 0, /* Min. Hoehe */
388 0, /* Max. Breite */
389 0, /* Max. Hoehe */
390 WBENCHSCREEN /* Screen-Typ */
391 };
392
393 BOOL ende;
394
395
396 main()
397 {
398 ULONG MessageClass;
399 USHORT code;
400
401 /* PropInfo der RGB-Gadgets setzen */
402 r_prop.Flags = g_prop.Flags = b_prop.Flags = FREEHORIZ!AUTOKNOB;
403 r_prop.HorizBody = g_prop.HorizBody = b_prop.HorizBody = 0x1000;
404 r_prop.HorizPot = g_prop.HorizPot = b_prop.HorizPot = 0x0000;
405
406 if(!(GfxBase = (struct GfxBase *)OpenLibrary("graphics.library",0)))
407 {
408 /* Grafik-Bibliothek oeffnen */
409 close_things();
410 exit();
411 }
412
413 /* Intuition oeffnen */
414 if(!(IntuitionBase = (struct IntuitionBase *)
415 OpenLibrary("intuition.library",0)))
416 {
417 close_things();
418 exit();
419 }
420
421 if (!(w = (struct Window *)OpenWindow(&nw))) /* Window oeffnen */
422 {
423 close_things();
424 exit();
425 }
426
427 rp = w->RPort;
428 screen = w->WScreen;
429
430 SetRGB4(&screen->ViewPort,0,0,0,0); /* Hintergrundfarbe setzen */
431
432 RefreshGadgets(&bool12_gad,w,NULL); /* Gadgets zeichnen */
433
434 ende = FALSE;
435
436 for(;;) /* Endlosschleife */
437 {
438 if (message = (struct IntuiMessage *)GetMsg(w->UserPort))
439 {
440 /* Message empfangen und verarbeiten */
441 MessageClass = message->Class; /* Message retten */
442 code = message->Code;
443 ReplyMsg(message); /* Message beantworten */
444 switch (MessageClass)
445 {
446 case GADGETUP :
447 case GADGETDOWN : do_gadgets(message, w);
448 break; /* Wenn Gadget betaetigt, dann...*/
449 case MOUSEBUTTONS: break;
450 }
451 }
452 }
453
454 if (ende == TRUE)

```

```
449 {
450 close_things();
451 exit();
452 };
453 }
454 }
455
456
457 do_gadgets (mes, win) /* Gadgets ausfuehren */
458 struct IntuiMessage *mes;
459 struct Window *win;
460 {
461 struct Gadget *igad;
462 int gadgid;
463 ULONG val;
464
465 igad = (struct Gadget *) mes->IAddress; /* Ptr auf ein Gadget
466 gadgid = igad->GadgetID; /* Eigene Identitaetsnummer */
467 val = (ULONG)TexString.LongInt;
468 switch(gadgid)
469 {
470 /* Farben aendern */
471 case 0: SetRGB4(&screen->ViewPort,0,r_prop.HorizPot/4096,
472 g_prop.HorizPot/4096,b_prop.HorizPot/4096);
473 break;
474 case 1: SetRGB4(&screen->ViewPort,0,r_prop.HorizPot/4096,
475 g_prop.HorizPot/4096,b_prop.HorizPot/4096);
476 break;
477 case 2: SetRGB4(&screen->ViewPort,0,r_prop.HorizPot/4096,
478 g_prop.HorizPot/4096,b_prop.HorizPot/4096);
479 break;
480 case 3: break;
481 case 5: ende = TRUE;
482 break;
483 case 6: DisplayBeep(NULL);
484 break;
485 };
486
487
488 close_things() /* Unterfunktion zum Schliessen */
489 {
490 CloseWindow(w); /* Libs und Window schliessen */
491 CloseLibrary(GfxBase);
492 CloseLibrary(IntuitionBase);
493 }
```

# System-Meldungen

System-Meldungen werden dazu benutzt, dem Anwender wichtige Informationen z.B. über das Amiga-System zu vermitteln oder um Entscheidungen vom Anwender zu verlangen. Hierbei finden zwei verschiedene Typen von Systemmeldungen Verwendung:

Dies ist zunächst einmal die Verwendung von Alerts (englisch Alarm). Dies sind rot-blinkende Warnmeldungen des Systems, die immer in einer horizontalen Auflösung von 640 Pixels dargestellt werden. Wenn ein Alert nur einen Teil des Bildschirms nach unten schiebt, so ist es durchaus möglich, daß man in das Programm zurückkehrt. Ist der ganze Bildschirm schwarz gefärbt, und an der obersten Stelle blinkt das Alert, so bedeutet dies den totalen Systemabsturz und das erneute Einlegen der Workbench-Disk.

Eine weitere Möglichkeit stellen die System-Requester dar. Sie tauchen nur dann auf, wenn Entscheidungen oder weitere Informationen vom Benutzer verlangt werden. Sie bestehen aus einem Window, in dem sich verschiedene Gadgets befinden.

## 9.1 Die Alerts

Wie in Kapitel 9 schon erwähnt, sind Alerts rot-blinkende »Hilferufe« des Systems in einer Auflösung von 640 Pixels auf einem schwarzem Hintergrund. Diese Hilferufe können dann durch Betätigen einer Maustaste quittiert werden.

Je nachdem, ob es ein »DeadEnd-Alert« oder ein »Recovery-Alert« ist, gelangt man zurück zum Programm oder auch nicht!

### »DeadEnd-Alerts«

DeadEnd-Alerts (englisch Sackgasse) signalisieren den kompletten Systemabsturz. Sie sind daran zu erkennen, daß nur das jeweilige Alert auf dem schwarzen Bildschirm blinkt. Hier können Sie unternehmen, was Sie wollen, wenn Sie eine Maustaste betätigen, bedeutet dies immer den Neustart mit der Workbench!

### »Recovery-Alerts«

Diese Art von Alerts, die sogenannten rückkehrfähigen Alerts, teilen dem Benutzer mit, daß er knapp an einem Systemabsturz vorbeigekommen ist. Erkennbar sind diese Alerts daran, daß der augenblickliche Bildschirm nach unten geschoben wird und ein roter Kasten auf schwarzem Hintergrund am oberen Teil des Bildschirms blinkt. Drücken Sie nun eine Maustaste, so wird das System nicht zurückgesetzt. Wenn jedoch das Amiga-System anschließend nicht mehr mit der Arbeit fortfahren kann, kann es durchaus vorkommen, daß es »vergißt« nach dem Anzeigen des Alerts in das Programm zurückzukehren und das System trotzdem zurücksetzt.

Beim Aufrufen von Alerts besteht einmal die Möglichkeit vorbereitete Alerts von »Exec« oder selbst-definierte Alerts von »Intuition« aus aufzurufen. Sehr reizvoll sind hier die selbst-definierten Alerts, da der Programmierer über sie wichtige Informationen bzw. Meldungen sehr einfach und wirkungsvoll dem Programm-Anwender mitteilen kann.

### 9.1.1 Der Aufruf von System-Alerts

System-Alerts sind vorbereitete Alerts, die einen festen Text besitzen. Für das Aufrufen von System-Alerts besitzt das Exec-Library einen bestimmten Befehl:

`Alert(AlertNummer, Parameter)`

AlertNummer gibt die Nummer des Alerts an. Eine Übersicht über die vorbereiteten Alerts finden Sie im Anhang C. Die Variable »Parameter« kann im Normalfall auf 0 gesetzt werden.

```
1 /*****
2
3 Exec-Alert-Demo
4 last update 26/05/87
5 von Joerg Koch und Frank Kremser
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Diese Demonstration gibt zwei System-Alerts aus. Der Erste kehrt wieder
11 zurueck. Der Zweite ist allerdings ein Dead-End-Alert.
12
13 *****/
14
15 #include <exec/alerts.h> /* Include-Files einlesen */
16 #include <exec/types.h>
17 #include <exec/tasks.h>
18 #include <exec/libraries.h>
19 #include <exec/devices.h>
20 #include <exec/execbase.h>
21 #include <hardware/blit.h>
22 #include <devices/keymap.h>
23 #include <graphics/regions.h>
24 #include <graphics/copper.h>
25 #include <graphics/gels.h>
26 #include <graphics/gfxbase.h>
27 #include <graphics/gfx.h>
28 #include <graphics/clip.h>
29 #include <graphics/view.h>
30 #include <graphics/rastport.h>
31 #include <graphics/layers.h>
32 #include <intuition/intuition.h>
33
34
35 struct ExecBase *ExecBase;
36
37
38 main()
39 {
40 LONG warte;
41
42 if ((ExecBase = (struct ExecBase *) /* Exec-Bibliothek oeffnen */
43 OpenLibrary("exec.library", 0)) == 0) exit();
44
45 Alert(AN_BitMap,0,0); /* Erster Alert */
46
47 for(warte = 0; warte < 100000; warte++); /* Abwarten */
48
49 Alert(AT_DeadEnd + AG_NoMemory + AO_BootStrap,0,0); /* Zweiter Alert */
50
51 CloseLibrary(ExecBase); /* Exec-Bibliothek schliessen */
52 }
```

## 9.1.2 Der Aufruf von Intuition-Alert

Intuition-Alerts sind komfortabler als Exec-Alerts, denn bei ihnen kann der Programmierer den Text und die Größe selbst bestimmen. Aufgerufen werden Intuition-Alerts mit:

`DisplayAlert(Type, Nachricht, Höhe)`

Die Variable »Type« gibt an, ob nach dem Betätigen der Maustaste mit dem Programm fortgefahren werden soll oder ob das System zurückgesetzt wird. Ist in Type `RECOVERY_ALERT` eingetragen, so kann nach dem Anzeigen des Alerts das Programm fortfahren. Wenn das System zurückgesetzt werden soll, muß für Type `DEADEND_ALERT` eingetragen werden.

»Nachricht« ist der Pointer auf eine Nachricht. Die Nachricht muß wie folgt aufgebaut sein:

Zuerst muß die Position des Textes durch eine 16-Bit x- und eine 8-Bit y-Koordinate festgelegt werden. Danach folgt der Text, der mit einer 0 enden muß, damit Intuition erkennt, wann der Text zu Ende ist. Der letzte Parameter des Texts gibt an, ob ein weiterer Text folgt. Folgt keine weitere Nachricht, so muß nach dem Text, der mit einer 0 endet, eine weitere 0 eingetragen werden. Wenn ein weiterer Text folgen soll, so muß der Parameter ungleich 0 sein.

Die Variable »Höhe« gibt an, wie hoch der rot-blinkende Kasten, in dem sich die Nachricht befindet, werden soll.

```

1 /*****
2
3 Alerts-Demonstration
4 last update 25/05/87
5 von Frank Kremser und Joerg Koch
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Diese Demonstration erzeugt zwei Alerts in Folge. Der erste kehrt
11 zurueck, der zweite bewirkt einen Neustart.
12
13 *****/
14
15 #include <exec/types.h> /* Einladen der Include-Files */
16 #include <exec/tasks.h>
17 #include <exec/libraries.h>
18 #include <exec/devices.h>
19 #include <devices/keymap.h>
20 #include <hardware/blit.h>
21 #include <graphics/regions.h>
22 #include <graphics/copper.h>
23 #include <graphics/gels.h>
24 #include <graphics/gfxbase.h>
25 #include <graphics/gfx.h>
26 #include <graphics/clip.h>
27 #include <graphics/view.h>
28 #include <graphics/rastport.h>
29 #include <graphics/layers.h>
30 #include <intuition/intuition.h>
31 #include <intuition/intuitionbase.h>
32
33
34 struct IntuitionBase *IntuitionBase;
35
36 char alert1[] = /* Text und Parameter fuer das Recovery-Alert */
37 {
38 "\0240\30Dies ist ein RECOVERY-Alert\01\0250\60Bitte druecken Sie eine Maustast
39 };
40
41 char alert2[] = /* Text und Parameter fuer das Dead-End-Alert */
42 {
43 "\0240\30Dies ist ein DEADEND-Alert\01\0250\60Bitte druecken Sie eine Maustaste
44 };
45
46
47 main()
48 {
49 LONG warte; /* Variable fuer Warte-Schleife */
50
51 if ((IntuitionBase = (struct IntuitionBase *) /* Intuition-Library */
52 OpenLibrary("intuition.library", 0)) == 0) exit(); /* oeffnen */
53
54 DisplayAlert(RECOVERY_ALERT, &alert1[0], 90); /* Erstes Alert dar- */
55 /* stellen */
56 for(warte = 0; warte < 300000; warte++); /* und warten */
57
58 DisplayAlert(DEADEND_ALERT, &alert2[0], 90); /* Zweites Alert */
59 /* darstellen */
60 CloseLibrary(IntuitionBase); /* Intuition-Library */
61 } /* schliessen */

```

## 9.2 Einfache Systemmeldungen durch Requester

Requester sind menüähnliche Fenster. Sie können Standard-Requester verwenden oder eigene definieren. Sie werden deshalb als Requester bezeichnet, weil der Benutzer erst die Nachfrage (Request) des Systems oder des Programms beantworten muß, bevor im Programm fortgefahren werden kann.

Die Nachfrage muß meistens durch das Anklicken eines Gadgets mit der Aufschrift »OK« oder »CANCEL« beantwortet werden. »OK« dient meistens zum Auslösen einer Funktion, während »CANCEL« die Nachfrage abbricht.

Das Fenster, aus dem das Requester besteht, ist nichts anderes als ein Window, somit kann es auch Front-, Schließ-, Back- und Größen-Gadgets besitzen. Es besitzt auch eine Titel-Leiste, an der es ergriffen und über den Bildschirm bewegt werden kann.

Dem Programmierer stehen unterschiedliche Methoden zur Verfügung, ein Requester erscheinen zu lassen. Die einfachste ist das Bilden eines Requesters mittels `AutoRequest()`, das die Abfrage wahr oder falsch erlaubt. Schwieriger sind dagegen die selbst-definierten Requester, die z.B. ein eigenes Image oder eigene Gadgets besitzen. Eine besondere Art Requester zu verwenden, bietet die Funktion `SetDMRequest()`. `SetDMRequest` läßt ein Requester erscheinen, wenn der Benutzer die Menü-Maustaste kurz hintereinander zweimal betätigt (Double-Klick).

Wichtig ist, daß jeder Requester zu einem Window gehört, in dem ein Programm abläuft. Aus diesem Grund muß auch bei fast jedem Requester-Befehl ein Zeiger auf die Window-Structure des Windows angegeben werden, zu dem der Requester gehören soll.

## 9.3 Die Requester-Structure

Möchte man von der vorgefertigten Form der Auto-Requester abweichen, so muß man eine eigene Requester-Structure anlegen. Diese hat folgende Form:

```
struct Requester
{
 struct Requester *OlderRequest;
 SHORT LeftEdge, TopEdge;
 SHORT Width, Height;
 SHORT RelLeft, RelTop;
 struct Gadget *ReqGadget;
 struct Border *ReqBorder;
 struct IntuiText *ReqText;
 USHORT Flags
 UBYTE BackFill;
 struct Layer *ReqLayer;
 UBYTE ReqPad1[32];
 struct BitMap *ImageBMap;
 struct Window *RWindow;
 UBYTE ReqPad2[36];
};
```

|                   |                                                                                                                                                                                                       |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OlderRequest      | Zeiger auf den zuletzt behandelten Requester (Wird von Intuition belegt).                                                                                                                             |
| LeftEdge, TopEdge | Hier muß die Position eingetragen werden, an der das Requester erscheinen soll.                                                                                                                       |
| Width, Height     | Width und Height geben die Größe des Requesters an.                                                                                                                                                   |
| RelLeft, RelTop   | Diese beiden Variablen geben die relative Position des darzustellenden Requesters in Bezug auf die augenblickliche Position des Mauszeigers an. Dazu muß das POINTREL-Flag bei Flags gesetzt sein.    |
| ReqGadget         | Hier muß der Zeiger auf das erste Gadget in der Gadgetliste eingetragen werden. Mindestens eines Gadgets muß das ENDGADGET-Flag gesetzt haben, damit das Requester anschließend wieder gelöscht wird. |
| ReqBorder         | Hier muß der Zeiger auf die Border-Struktur eingetragen werden, die das Requester umranden soll. Wird keine Border verwendet so ist diese Variable »NULL«.                                            |

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReqText  | ReqText gibt den Zeiger auf die Intuition-Text-Structure an, die den Text enthält, der in das Requester geschrieben werden soll. Sollen mehrere Texte verwendet werden, so muß in der IntuiText-Structure jeweils ein Zeiger auf den nächsten Text angegeben werden.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Flags    | <p>Durch das Setzen von verschiedenen Flags kann das Verhalten und Aussehen von Requestern bestimmt werden :</p> <p><b>POINTREL</b> Dies Flag gibt an, daß das Requester beim Initialisieren relativ zur augenblicklichen Mauszeiger-Position dargestellt werden soll. Bei der alten Workbench-version hat dieses Flag keine Bedeutung.</p> <p><b>PREDRAWN</b> Wenn eigene BitMaps verwendet werden sollen, muß dieses Flag gesetzt werden.</p> <p>Diese Flags werden von Intuition gesetzt:</p> <p><b>REQOFFWINDOW</b> Dieses Flag ist gesetzt, wenn das Requester aktiv, aber das Window des Requesters nicht aktiv ist.</p> <p><b>REQACTIVE</b> Dieses Flag gibt an, ob das Requester augenblicklich benutzt oder nicht benutzt wird.</p> <p><b>SYSREQUEST</b> Dieses Flag wird gesetzt, wenn das Requester ein System-Requester ist.</p> |
| BackFill | Hier wird die Farbe eingetragen, mit der der Hintergrund des Requesters ausgefüllt werden soll.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| ReqLayer | Enthält die Adresse der »Grafik-Schicht«, die das Requester verdeckt.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

ReqPad1[32]

Ist für die interne Benutzung reserviert.

ImageBMap

Wenn ein eigenes Bit-Map verwendet werden soll, muß hier der Zeiger auf das eigene Bit-Map eingetragen werden. Zudem muß noch in Flags das Flag PREDRAWN gesetzt sein. Soll kein eigenes Bit-Map verwendet werden, so muß hier »NULL« eingetragen werden.

RWindow

Dies ist eine System-Variable.

ReqPad2[36]

Ist für die interne Benutzung reserviert.

## 9.4 Selbstdefinierte Requester

Selbstdefinierte Requester sind Requester, die von Grund auf vom Programmierer selbst definiert werden. Im Vergleich zu `AutoRequest()` muß bei selbstdefinierten Requestern alles selbst in die Hand genommen werden. Zunächst muß eine Structure, die Requester-Structure, die wir im vorhergehenden Kapitel beschrieben haben, definiert werden. In dieser Structure braucht nur das Nötigste angegeben zu werden, da die Funktion `InitRequest()` alles Übrige erledigt, um eine ordentliche Requester-Structure zu definieren.

Durch anschließendes Aufrufen der Funktion `Request()` oder `SetDMRequest` wird Ihr Requester in dem jeweiligen Window dargestellt. Damit das Requester wieder gelöscht werden kann, muß eines der selbstdefinierten Gadgets das Flag `ENDGADGET` in der Variable »Activate« gesetzt haben, damit das Requester bei Betätigung dieses Gadgets vom Bildschirm verschwindet.

`SetDMRequest` hat fast dieselbe Funktion wie `Request()`. Der Unterschied liegt darin, daß `SetDMRequest` ein Double-Menü-Requester bildet, das nur dann sichtbar wird, wenn der Benutzer zweimal die Menütaste – gleich rechte Taste – der Maus betätigt.

Gelöscht werden kann das Requester, neben dem Setzen des Flags `ENDGADGET`, zusätzlich durch das Anwenden der Befehle `EndRequest()` und `ClearDMRequest`.

## 9.5 Das Auto/System-Request

Falls nur eine simple »JA« oder »NEIN«-Antwort und auch kein eigenes BitMap sowie Gadgets benötigt werden, reicht die Darstellung von Requesters durch die Intuition-Funktion `AutoRequest()` vollkommen aus. Wenn diese Funktion aufgerufen wird, bildet Intuition das Requester, stellt es mit dem gewünschten Text dar und wartet auf eine Antwort des Benutzers.

Der Text, der dargestellt werden soll, wird mit `AutoRequest()` übergeben. Dabei ist zu beachten, daß sich die positive Nachricht immer in der linken Ecke und die negative Nachricht in der rechten Ecke des Requesters befindet. Der positive Text kann auch auf »NULL« gesetzt werden, wenn dem Benutzer keine Auswahlmöglichkeit zwischen »Wahr« und »Falsch« gegeben werden soll, was jedoch relativ selten vorkommt.

Zu jedem der zwei Auswahlmöglichkeiten können IDCMP-Flags gesetzt werden. Dies sind bestimmte Flags, die die Kommunikation zwischen dem Anwender und dem System regeln. Wenn keine weiteren speziellen Flags benötigt werden, kann `PosFlags`, bzw. `NegFlags` gleich »NULL« gesetzt werden. Ansonsten können folgende Flags verwendet werden:

|                  |                                                                                                                                                        |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>REQSET</b>    | Wenn dieses Flag gesetzt ist, erhalten Sie eine Nachricht, wenn das Requester geöffnet ist.                                                            |
| <b>REQCLEAR</b>  | Wenn das letzte Requester aus der zugehörigen Window-Structure gelöscht und dieses Flag gesetzt ist, erhalten Sie eine Nachricht.                      |
| <b>REQVERIFY</b> | Dieses Flag kann gesetzt werden, um sicher zu gehen, daß andere wichtige Entscheidungen zuerst getroffen werden, bevor das Requester dargestellt wird. |

`AutoRequest()` gibt »TRUE« zurück, falls das positive Gadget betätigt wurde und »FALSE« für das negative Gadget.

Die `AutoRequest()`-Funktion ruft die Funktion `BuildSysRequest()` auf. Durch sie wird das Requester dargestellt. Die Gadgets, die durch `BuildSysRequest()` definiert werden, haben folgende Flags in ihrer Structure gesetzt:

**BOOLGADGET** für eine Abfrage ob »Wahr« oder »Falsch«,

**RELVERIFY** um sicherzugehen, daß der Anwender das Gadget auch wirklich angeklickt hat,

**REQUGADGET** um zu überprüfen, ob es sich um ein Requester-Gadget handelt,

**TOGGLESELECT**-Flag, das besagt, daß der Zustand des Gadgets durch Anklicken verändert wird.

Für den Programmierer haben diese Flags in diesem Fall keine Bedeutung, da sie von Intuition selbst kontrolliert werden.

## 9.6 Die Requester-Befehle

### 9.6.1 AutoRequest

|            |                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                           |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | <code>bool = AutoRequest(Window, TitelText, PosText, NegText, PosFlag, NegFlag, Breite, Höhe);</code>                                                                                                                                                                                                                   |                                                                                                                                                                                                                           |
| Funktion:  | AutoRequest bildet durch Aufrufen der Funktion <code>BuilSysRequest</code> einen Requester mit dem angegebenen positiven und negativen Text. Danach wartet es so lange, bis der jeweilige Text bestätigt wurde und gibt den Wert »TRUE« für Bestätigung des positiven Textes und »FALSE« für den negativen Text zurück. |                                                                                                                                                                                                                           |
| Parameter: | Window                                                                                                                                                                                                                                                                                                                  | -> Zeiger auf die Window-Structure des Windows, zu dem der Requester gehören soll                                                                                                                                         |
|            | TitelText                                                                                                                                                                                                                                                                                                               | -> Zeiger auf eine <code>IntuiText</code> – Structure. Dieser Text wird dann in die Titelleiste des Requesters geschrieben.                                                                                               |
|            | PosText                                                                                                                                                                                                                                                                                                                 | -> Zeiger auf die <code>IntuiText</code> -Structure des <code>NegText</code> positiven und negativen Textes.                                                                                                              |
|            | PosFlag                                                                                                                                                                                                                                                                                                                 | -> geben die Flags für die Gadgets der <code>NegFlag</code> positiven und negativen Nachricht an. Hier kann »NULL« eingetragen werden, da <code>AutoRequest</code> das Setzen der richtigen Flags selbstständig erledigt. |
|            | Breite,Höhe                                                                                                                                                                                                                                                                                                             | -> Dimension des Requesters.                                                                                                                                                                                              |
| Ergebnis:  | bool                                                                                                                                                                                                                                                                                                                    | -> ist bool gleich »TRUE«, so wurde die positive Nachricht gewählt, bei »FALSE« die negative.                                                                                                                             |
| Datentyp:  | <pre>struct Window *Window; struct IntuiText *TitelText, *PosText, *NegText; WORD PosFlag, NegFlag; int Breite, Höhe; bool bool;</pre>                                                                                                                                                                                  |                                                                                                                                                                                                                           |

- Sonstiges: Wird für die Window-Structure »NULL« gesetzt, so öffnet Intuition selbst ein Window.
- Referenz: Siehe auch BuildSysRequest

## 9.6.2 BuildSysRequest

- Syntax: `adr = BuildSysRequest(Window, TitelText, PosText, NegText, Flags, Weite, Höhe);`
- Funktion: BuildRequest bildet ein Requester mit dem angegebenen positiven und negativen Text.
- Parameter:
- Window -> Zeiger auf die Window-Structure des Windows, dem der Requester zugeordnet werden soll
  - TitelText -> Zeiger auf eine IntuiText-Structure. Dieser Text wird dann in die Titelleiste des Requesters geschrieben.
  - PosText -> Zeiger auf die IntuiText-Structure des positiven Textes
  - NegText -> Zeiger auf die IntuiText-Structure des negativen Textes
  - Flags -> IDCMPFlags der Window
  - Breite, Höhe -> Dimension des Requesters.
- Ergebnis: `adr` -> Zeiger auf die Window-Structure des Requesters.
- Datentyp: `struct Window *Window;`  
`struct IntuiText *TitelText, *PosText, *NegText;`  
`WORD Flags;`  
`int Breite, Höhe;`  
`ULONG adr;`
- Sonstiges: BuildSysRequest() initialisiert die IDCMP-Flags des Window, so daß der UserPort oder WindowPort nur mit Wait() abgefragt werden muß, ob die Flags übereinstimmen.
- Ist Window gleich »NULL«, so bildet Intuition selbst ein Window.
- Referenz: Siehe auch AutoRequest

### 9.6.3 ClearDMRequest

- Syntax:** fertig = ClearDMRequest(Window);
- Funktion:** Löscht das jeweilige Double-Menü-Requester aus der Requester-Liste des angegebenen Windows.
- Parameter:** Window -> Zeiger auf die Window-Structure des Windows, in dessen Requester-Liste sich das Double-Menü-Requester befindet.
- Ergebnis:** fertig -> TRUE, wenn es gelöscht werden konnte, sonst FALSE.
- Datentyp:** struct Window \*Window;  
bool fertig;
- Sonstiges:** Gesetzt wird das DM-Requester mit SetDMRequest.
- Referenz:** Siehe auch SetDMRequest

### 9.6.4 EndRequest

- Syntax:** EndRequest(Requester, Window);
- Funktion:** Löscht das spezifizierte Requester aus der Requester-Liste des angegebenen Windows.
- Parameter:** Requester -> Zeiger auf die Requester-Structure des Requesters, das gelöscht werden soll.  
Window -> Zeiger auf die Window-Structure des Windows, in dessen Requester-Liste sich der Requester befindet.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct Requester \*Requester;  
struct Window \*Window;
- Sonstiges:** Ein Requester kann mit Request() gesetzt werden.
- Referenz:** Siehe auch Request

### 9.6.5 FreeSysRequest

- Syntax:** FreeSysRequest(Window);
- Funktion:** Löscht alle Requester aus der Requester-Liste des angegebenen Windows, die mit BuildSysRequest erstellt wurden.

|            |                                                                  |                                                                                                       |
|------------|------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| Parameter: | Window                                                           | -> Zeiger auf die Window-Structure des Windows, in dessen Requester-Liste sich die Requester befinden |
| Ergebnis:  | Kein Ergebnis.                                                   |                                                                                                       |
| Datentyp:  | struct Window *Window;                                           |                                                                                                       |
| Sonstiges: | Ein solcher Requester kann mit BuildSysRequest() gesetzt werden. |                                                                                                       |
| Referenz:  | Siehe auch BuildSysRequest                                       |                                                                                                       |

### 9.6.6 InitRequest

|            |                                                                                                                                                                                               |                                                            |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| Syntax:    | InitRequest(Requester);                                                                                                                                                                       |                                                            |
| Funktion:  | Initialisiert eine Requester-Structure.                                                                                                                                                       |                                                            |
| Parameter: | Requester                                                                                                                                                                                     | -> Zeiger auf die zu initialisierende Requester-Structure. |
| Ergebnis:  | Kein Ergebnis.                                                                                                                                                                                |                                                            |
| Datentyp:  | struct Requester *Requester;                                                                                                                                                                  |                                                            |
| Sonstiges: | Bevor ein Requester initialisiert werden kann, muß die dazugehörige Gadget-Structure erstellt sein. Diese Funktion bringt das Requester nicht auf den Screen, dafür ist Request() vorgesehen. |                                                            |
| Referenz:  | Siehe auch Request                                                                                                                                                                            |                                                            |

### 9.6.7 Request

|            |                                                        |                                                                                                       |
|------------|--------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| Syntax:    | bool = Request(Requester, Window);                     |                                                                                                       |
| Funktion:  | Stellt das spezifizierte Requester auf dem Screen dar. |                                                                                                       |
| Parameter: | Requester                                              | -> Zeiger auf die Requester-Structure des Requesters, der dargestellt werden soll.                    |
|            | Window                                                 | -> Zeiger auf die Window-Structure des Windows, zu dessen Requester-Liste das Requester gehören soll. |
| Ergebnis:  | bool                                                   | -> ist »TRUE«, falls der Requester dargestellt werden konnte und »FALSE«, wenn nicht.                 |

- Datentyp:** struct Requester \*Requester;  
struct Window \*Window;  
bool bool;
- Sonstiges:** Bevor man das Requester mit Request() darstellen kann, muß es zuvor mit InitRequest initialisiert worden sein.  
  
Diese Routine ignoriert das REQVERIFY Flag in der Window-Structure.
- Referenz:** Siehe auch InitRequest

### 9.6.8 SetDMRequest

- Syntax:** bool = SetDMRequest(Window, DMRequester);
- Funktion:** Stellt das spezifizierte Requester als Double-Menü-Requester auf dem Screen dar.
- Parameter:** Requester -> Zeiger auf die Requester-Structure des Requesters, der dargestellt werden soll.  
  
Window -> Zeiger auf die Window-Structure des Windows, zu dessen Requester-Liste das Requester gehören soll.
- Ergebnis:** bool -> falls nicht schon ein DM-Requester installiert ist, wird das neue Requester installiert und »TRUE« zurückgegeben, andernfalls »FALSE«.
- Datentyp:** struct Requester \*Requester;  
struct Window \*Window;  
bool bool;
- Sonstiges:** DMRequester sind besondere Requester, die auf einen Double-Klick des Menü-Knopfes der Maus hin erscheinen.
- Referenz:** Siehe auch ClearDMRequest

```

1 /*****
2
3 Auto-Requester-Demonstration
4 last update 26/05/87
5 von Joerg Koch und Frank Kremser
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Diese Demonstration erzeugt ein einfaches Requester durch die Funktion
11 AutoRequest
12
13 *****/
14
15 #include <exec/types.h> /* Include-Files einladen */
16 #include <exec/nodes.h>
17 #include <exec/lists.h>
18 #include <exec/ports.h>
19 #include <exec/devices.h>
20 #include <devices/keymap.h>
21 #include <graphics/regions.h>
22 #include <graphics/copper.h>
23 #include <graphics/gels.h>
24 #include <graphics/gfxbase.h>
25 #include <graphics/gfx.h>
26 #include <graphics/clip.h>
27 #include <graphics/view.h>
28 #include <graphics/rastport.h>
29 #include <graphics/layers.h>
30 #include <intuition/intuition.h>
31 #include <hardware/blit.h>
32
33 struct IntuitionBase *IntuitionBase; /* Lib-Zeiger */
34 struct GfxBase *GfxBase;
35
36 struct IntuiText text = /* Text fuer das Requester */
37 { /* erstellen */
38 0, /* DetailPen */
39 1, /* BlockPen */
40 JAM1, /* Drawmode */
41 17, /* X-Wert relativ */
42 20, /* Y-Wert relativ */
43 NULL, /* spez. Zeichensatz */
44 "Soll das Programm beendet werden?", /* Text */
45 NULL, /* naechster Text */
46 };
47
48 struct IntuiText postext = /* Text fuer das positive */
49 { /* Gadget */
50 0,
51 1,
52 JAM2,
53 7,
54 4,
55 NULL,
56 "Ja!",
57 NULL,
58 };
59
60 struct IntuiText negtext = /* Text fuer das negative */
61 { /* Gadget */
62 0,
63 1,
64 JAM2,

```

```
65 7,
66 4,
67 NULL,
68 "Nein!",
69 NULL
70 };
71
72
73 main()
74 {
75 BOOL ergebnis;
76 /* Graphik-Bibliothek oeffnen */
77 if(!(GfxBase = (struct GfxBase *)OpenLibrary("graphics.library",0)))
78 exit();
79 /* Intuition-Bibliothek oeffnen */
80 if(!(IntuitionBase = (struct IntuitionBase *)
81 OpenLibrary("intuition.library",0)))
82 exit();
83
84 for(;;) /* Endlosschleife */
85 {
86 ergebnis = AutoRequest(NULL,&text,&postext,&negtext,
87 NULL,NULL,320,70); /* Requester darstellen */
88 if (ergebnis == TRUE) /* Wenn das positive */
89 { /* Gadget betaetigt */
90 CloseLibrary(GfxBase); /* wurde -> abbrechen */
91 CloseLibrary(IntuitionBase); /* Libs schliessen */
92 exit();
93 };
94 };
95 }
```

```
1 /*****
2
3 Double-Menu-Requester-Demonstration
4 last update 26/05/87
5 von Joerg Koch und Frank Kremser
6 (c) Markt & Technik 1987
7 *****/
8
9 Diese Demonstration erzeugt einen Double-Menu-Requester, der erst durch
10 zweimaliges Betaetigen der rechten Maustaste sichtbar wird.
11
12 *****/
13
14 #include <exec/types.h> /* Include-Files einlesen */
15 #include <exec/nodes.h>
16 #include <exec/lists.h>
17 #include <exec/ports.h>
18 #include <exec/devices.h>
19 #include <devices/keymap.h>
20 #include <graphics/regions.h>
21 #include <graphics/copper.h>
22 #include <graphics/gels.h>
23 #include <graphics/gfxbase.h>
24 #include <graphics/gfx.h>
25 #include <graphics/clip.h>
26 #include <graphics/view.h>
27 #include <graphics/rastport.h>
28 #include <graphics/layers.h>
29 #include <intuition/intuition.h>
30 #include <hardware/blit.h>
```

```

31
32 struct IntuitionBase *IntuitionBase;
33 struct GfxBase *GfxBase;
34 struct Window *w;
35
36 struct IntuiText postext = /* Text fuer das positive Gadget */
37 {
38 1, /* erstellen */
39 3, /* DetailPen */
40 JAM2, /* BlockPen */
41 7, /* Drawmode */
42 2, /* Linke Ecke */
43 2, /* Obere Ecke */
44 NULL, /* System-Zeichensatz */
45 " OK ", /* Text */
46 NULL, /* Zeiger auf naechsten Text */
47 };
48
49 SHORT Pairs[] = /* Koordinatenpaare fuer eine */
50 {
51 0, 0, /* Border definieren */
52 51, 0,
53 51, 12,
54 0, 12,
55 0, 0
56 };
57
58 struct Border butt_border = /* Border-Structure fuer Rand */
59 {
60 -1, /* um Gadget */
61 -1, /* Linke Ecke relativ */
62 2, /* Obere Ecke relativ */
63 2, /* DetailPen */
64 0, /* BlockPen */
65 JAM1, /* Drawmode */
66 5, /* Anzahl der Koordinaten-Paare */
67 (SHORT *) Pairs, /* Zeiger auf die Koordinaten */
68 NULL, /* Zeiger auf die naechste Border */
69 };
70
71 struct Gadget gad = /* Gadget erstellen */
72 {
73 NULL, /* Naechstes Gadget */
74 30, /* Linke Ecke */
75 46, /* Obere Ecke */
76 50, /* Breite */
77 11, /* Hoehe */
78 GADGHCOMP, /* Flags */
79 RELVERIFY|GADGIMMEDIATE|ENDGADGET, /* Activation */
80 BOOLGADGET|REGGADGET, /* Gadget-Typ */
81 (APTR)&butt_border, /* GadgetRender - Border */
82 NULL, /* SelectRender - Border */
83 &postext, /* Gadget-Text */
84 0, /* MutualExclude */
85 0, /* SpecialInfo */
86 0, /* Gadget-Identitaet */
87 0, /* User-Daten */
88 };
89
90 struct IntuiText text =
91 {
92 1,
93 3,
94 JAM2,
95 13,
96 5,

```

```
95 0,
96 "Requester",
97 0
98);
99
100 SHORT ReqPairs[] =
101 {
102 5, 3,
103 95, 3,
104 95, 78,
105 5, 78,
106 5, 3,
107 };
108
109 struct Border out_border =
110 {
111 -1,
112 -1,
113 2,
114 0,
115 JAM1,
116 5,
117 (SHORT *) ReqPairs,
118 NULL
119 };
120
121 struct Requester request = /* Requester erstellen */
122 {
123 NULL, /* Older Requester */
124 50, /* Linke Ecke */
125 50, /* Obere Ecke */
126 100, /* Breite */
127 80, /* Hoehe */
128 0, /* RelLeft */
129 0, /* RelTop */
130 &gad, /* Requester-Gadget */
131 &out_border, /* Requester-Border */
132 &text, /* Requester-Text */
133 NULL, /* Flags */
134 3, /* Hintergrundfarbe */
135 NULL, /* Layer */
136 NULL, /* nur fuer das System */
137 NULL, /* Zeiger auf eigenes Requester-Image */
138 NULL, /* nur fuer das System */
139 NULL /* nur fuer das System */
140 };
141
142 struct NewWindow nw = /* Window erstellen */
143 {
144 0, /* Linke Ecke */
145 0, /* Obere Ecke */
146 640, /* Breite */
147 256, /* Hoehe */
148 0, /* DetailPen */
149 1, /* BlockPen */
150 REGCLEAR, /* IDCMPFlags */
151 ACTIVATE:WINDOWDEPTH:WINDOWDRAG, /* Flags */
152 NULL, /* Zeiger auf das erste Gadget */
153 NULL, /* Checkmark */
154 NULL, /* Window-Titel */
155 NULL, /* Zeiger auf den Screen */
156 NULL, /* Zeiger auf SuperBitmap */
157 0, /* Minimale Breite */
158 0, /* Minimale Hoehe */
```

```

159 0, /* Maximale Breite */
160 0, /* Maximale Hoehe */
161 WBENCHSCREEN /* Screen-Typ */
162 };
163
164
165 main()
166 {
167 struct IntuiMessage *message;
168 ULONG MessageClass;
169 USHORT code;
170
171 /* Grafik-Bibliothek oeffnen */
172 GfxBase = (struct GfxBase *)OpenLibrary("graphics.library",0);
173 if(GfxBase == NULL) exit();
174 /* Intuition-Bibliothek oeffnen */
175 IntuitionBase = (struct IntuitionBase *)
176 OpenLibrary("intuition.library",0);
177 if(IntuitionBase == NULL) exit();
178
179 w = (struct Window *)OpenWindow(&nw); /* Window oeffnen */
180 if(w == NULL) exit();
181
182 SetDMRequest(w,&request); /* Dieser Befehl setzt ein Double-Menu-
183 Requester, das aktiviert werden muss,
184 indem zweimal die rechte Maustaste
185 betaetigt wird.
186 Soll ein normaler Requester verwendet
187 werden, so muss an dieser Stelle
188 folgender Befehl eingesetzt werden:
189
190 Request(&request,w); */
191 for(;;) /* Endlosschleife */
192 {
193 if(message = (struct IntuiMessage *)GetMsg(w->UserPort))
194 { /* Meldungen vom Port holen, */
195 MessageClass = message->Class; /* Message retten */
196 code = message->Code;
197 ReplyMsg(message); /* beantworten */
198 if(MessageClass == REQCLEAR) /* und auswerten */
199 { /* Wenn kein Requester mehr */
200 CloseWindow(w); /* vorhanden ist, soll dass */
201 CloseLibrary(GfxBase); /* Programm beendet werden */
202 CloseLibrary(IntuitionBase);
203 exit();
204 }
205 }
206 }

```



# **Die Ein- und Ausgabe**

Die Ein-/Ausgabe mittels DOS-Befehlen ist äußerst einfach zu handhaben, wie aus den Demonstrationsprogrammen ersichtlich ist.

Aber auch über die sogenannten Devices läßt sich die Ein- und Ausgabe steuern, womit der Programmierer erheblich größere Freiheiten hat, als bei den DOS-Befehlen, die ihn doch etwas einschränken.

## 10.1 DOS-Funktionen in Programmen

Auch von Programmen aus kann überaus leicht auf Daten zugegriffen werden, die sich auf Diskette befinden. Die nachfolgenden Befehle zeigen dies. Diese Befehle lassen sich aber nicht nur auf Disketten-Daten anwenden, sondern ermöglichen auch die Kommunikation mit anderen Geräten, wie zum Beispiel mit dem Drucker oder Modem, da die Ein- und Ausgabe leicht umgeleitet werden kann. Dies geschieht wie folgt:

```
datei = Open("PRT:",MODE_NEWFILE);
```

Anschließend können Daten mit dem Write-Befehl auf dem Drucker ausgegeben werden.

Überaus leicht können auch Daten in System-Windows unter Intuition ausgegeben werden, was allerdings für normale Anwendungen nicht sehr interessant sein dürfte. Sollte Sie dieses Thema interessieren, sehen Sie unter dem DOS-Befehl Execute nach.

Eine Anwendung, die wir an dieser Stelle schon beschreiben möchten, kann für Sie als Programmierer von sehr großer Bedeutung sein: Das Laden von Bilddateien in einen bestehenden Screen.

Zuerst muß ein normaler Screen geöffnet werden, wie es in Kapitel 2 beschrieben ist. Wenn der Screen das Format 320 x 256 Pixel hat, berechnet sich die Länge einer Bitplane wie folgt:

```
planelänge = 320 * 256 / 8;
```

Nehmen wir an, daß der geöffnete Screen fünf Bitplanes besitzt, so müssen diese fünf Bitplanes auch einzeln eingeladen werden. Im Programm sieht das Ganze dann folgendermaßen aus:

```
struct FileHandle *datei;
.
.
main()
{
 LONG länge;
 LONG erstesbyte;
 .
 .
 planelänge = 320 * 256 / 8;
 datei = Open("bild",MODE_OLDFILE);
 if (datei != 0)
 {
```

```

länge = Read(datei,erstesbyte,planelänge);/*anfl bis 5 sind */
länge = Read(datei,erstesbyte,planelänge);/*die Adressen der*/
länge = Read(datei,erstesbyte,planelänge);/*Bit-Plane-Puffer*/
länge = Read(datei,erstesbyte,planelänge);
länge = Read(datei,erstesbyte,planelänge);
Close(datei);
};
.
.
}

```

ScreenPtr muß dabei der Zeiger auf die Screen-Structure sein, die von OpenScreen zurückgegeben wird.

### 10.1.1 Close

Syntax: Close(datei);

Funktion: Schließt eine geöffnete Datei.

Parameter: datei -> Zeiger auf die FileHandle-Structure der Datei, die zuvor mit Open geöffnet worden sein muß.

Ergebnis: Kein Ergebnis.

Datentyp: struct FileHandle \*datei;

Sonstiges: Jede Datei, die zuvor mit Open geöffnet worden ist, muß, wenn sie nicht mehr benötigt wird, wieder geschlossen werden, da sonst ein IO-Fehler auftreten kann.

Zudem sollten Sie in Ihren Programmen nie Dateien schließen, die an anderer Stelle geöffnet worden sind.

Referenz: Siehe auch Open

### 10.1.2 CreateDir

Syntax: lock = CreateDir(name);

Funktion: Erzeugt ein neues Dateiverzeichnis auf der Diskette.

Parameter: name -> Name des neuen Datei-Unterverzeichnisses.

Ergebnis: lock -> Zeiger auf die Lock-Structure des Datei-Unterverzeichnisses.

Datentyp: char name[];  
struct Lock \*lock;

**Sonstiges:** In einer Lock-Structure sind alle wichtigen Informationen über eine bestimmte Datei bzw. über ein bestimmtes Datei-Verzeichnis enthalten, die DOS benötigt, um damit arbeiten zu können.

**Referenz:** Siehe auch CLI-Befehl `makedir` und `DeleteFile`

### 10.1.3 CurrentDir

**Syntax:** `altlock = CurrentDir(lock);`

**Funktion:** Wählt ein neues Datei-Verzeichnis als Arbeitsverzeichnis.

**Parameter:** `lock` -> Zeiger auf die Lock-Structure des Datei-Verzeichnisses, das gesetzt werden soll.

**Ergebnis:** `altlock` -> Zeiger auf die Lock-Structure des zuvor gesetzten Datei-Verzeichnisses.

**Datentyp:** `struct Lock *lock, *altlock;`

**Sonstiges:** Wird 0 zurückgegeben, bedeutet dies, daß das bisherige Verzeichnis das Hauptverzeichnis der Boot-Diskette war.

**Referenz:** Siehe auch CLI-Befehl `cd` und `Lock`

### 10.1.4 DeleteFile

**Syntax:** `erfolg = DeleteFile(name);`

**Funktion:** Löscht die spezifizierte Datei oder das spezifizierte Datei-Verzeichnis.

**Parameter:** `name` -> Name der zu löschenden Datei bzw. des zu löschenden Datei-Verzeichnisses.

**Ergebnis:** `erfolg` -> ist ein Wahrheitswert. Er ist TRUE, wenn gelöscht werden konnte, ansonsten ist er FALSE.

**Datentyp:** `char name[];`  
`bool erfolg;`

**Sonstiges:** Ein Datei-Verzeichnis kann nur dann gelöscht werden, wenn es keine weiteren Dateien oder Verzeichnisse mehr enthält.

**Referenz:** Siehe auch CLI-Befehl `delete`

## 10.1.5 DupLock

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                             |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| Syntax:    | neulock = DupLock(lock);                                                                                                                                                                                                                                                                                                                                                                                                                          |                                             |
| Funktion:  | Erzeugt eine Kopie der angegebenen Lock-Structure.                                                                                                                                                                                                                                                                                                                                                                                                |                                             |
| Parameter: | lock                                                                                                                                                                                                                                                                                                                                                                                                                                              | -> Zeiger auf zu kopierende Lock-Structure. |
| Ergebnis:  | neulock                                                                                                                                                                                                                                                                                                                                                                                                                                           | -> Zeiger auf Kopie der Lock-Structure.     |
| Datentyp:  | struct Lock *lock, *neulock;                                                                                                                                                                                                                                                                                                                                                                                                                      |                                             |
| Sonstiges: | Dieser Befehl wird verwendet, um mehrere Lock-Structures einer Datei bzw. eines Verzeichnisses zu erhalten.<br><br>Benötigt wird dies, wenn auf eine Datei bzw. auf ein Verzeichnis unabhängig voneinander zugegriffen werden soll. Möglich ist dies allerdings nur, wenn es eine READ-Lock-Structure ist (siehe Lock). Sollte trotzdem versucht werden, eine WRITE-Lock-Structure zu kopieren, tritt ein Fehler auf, und "neulock" ist gleich 0. |                                             |
| Referenz:  | Siehe auch Lock                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                             |

## 10.1.6 Examine

|            |                                                                                                                                                                                    |                                                                                                         |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|
| Syntax:    | erfolg = Examine(lock,puffer);                                                                                                                                                     |                                                                                                         |
| Funktion:  | Ermittelt den ersten Eintrag eines Verzeichnisses oder Detail-Informationen über eine Datei.                                                                                       |                                                                                                         |
| Parameter: | lock                                                                                                                                                                               | -> Zeiger auf die Lock-Structure der Datei bzw. eines Verzeichnisses.                                   |
|            | puffer                                                                                                                                                                             | -> Zeiger auf den Speicherbereich, in den der Eintrag bzw. die Informationen eingetragen werden sollen. |
| Ergebnis:  | erfolg                                                                                                                                                                             | -> TRUE, wenn kein Fehler aufgetreten ist.                                                              |
| Datentyp:  | struct Lock *lock;<br>struct FileInfoBlock *puffer;<br>bool erfolg;                                                                                                                |                                                                                                         |
| Sonstiges: | Wird ein Datei-Verzeichnis abgefragt, enthält der Puffer den Namen, die Größe und das Erstellungsdatum des Eintrages. Zudem kann ausgelesen werden, ob der Eintrag eine Datei oder |                                                                                                         |

ein Unterverzeichnis ist. Hier gilt: negativer Wert bedeutet Datei, positiver Wert Dateiverzeichnis.

Referenz: Siehe auch ExNext

### 10.1.7 Execute

Syntax: `erfolg = Execute(befehl, eingabe, ausgabe);`

Funktion: Führt einen CLI-Befehl aus.

Parameter: `befehl` -> CLI-Befehl, der ausgeführt werden soll.  
`eingabe` -> Zeiger auf die FileHandle-Structure der Eingabe-Einheit.  
`ausgabe` -> Zeiger auf die FileHandle-Structure der Ausgabe-Einheit.

Ergebnis: `erfolg` -> ist TRUE, wenn kein Fehler aufgetreten ist. Ansonsten ist er FALSE.

Datentyp: `char befehl[];`  
`struct FileHandle *eingabe, *ausgabe;`  
`bool ergebnis;`

Sonstiges: »befehl« muß den gesamten CLI-Befehl mit Parametern enthalten:

»rename test, endversion«

»eingabe« ist im Normalfall gleich 0.

Für »ausgabe« gleich 0 werden alle Meldungen des CLI-Befehls im aktuellen Window ausgegeben. Es kann aber auch eine andere Ausgabereinheit angegeben werden. Diese muß dann zuerst mit Open geöffnet werden. Der von Open zurückgegebene Zeiger auf die zugehörige FileHandle-Structure muß dann für »ausgabe« angegeben werden.

## 10.1.8 ExNext

|            |                                                                                                   |                                                                                                          |
|------------|---------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| Syntax:    | erfolg = ExNext(lock,puffer);                                                                     |                                                                                                          |
| Funktion:  | Ermittelt den Eintrag, der dem mit Examine ermittelten Eintrag folgt.                             |                                                                                                          |
| Parameter: | lock                                                                                              | -> Zeiger auf die Lock-Structure der Datei, bzw. eines Verzeichnisses.                                   |
|            | puffer                                                                                            | -> Zeiger auf den Speicherbereich, in den der Eintrag, bzw. die Informationen eingetragen werden sollen. |
| Ergebnis:  | erfolg                                                                                            | -> TRUE, wenn kein Fehler aufgetreten ist.                                                               |
| Datentyp:  | struct Lock *lock;<br>ULONG puffer;<br>bool erfolg;                                               |                                                                                                          |
| Sonstiges: | Sollten keine Einträge mehr vorhanden sein, enthält der Puffer den Eintrag ERROR_NO_MORE_ENTRIES. |                                                                                                          |
| Referenz:  | Für nähere Informationen siehe Examine.                                                           |                                                                                                          |

## 10.1.9 Info

|            |                                                                                     |                                                                                                                  |
|------------|-------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| Syntax:    | erfolg = Info(lock,info_data);                                                      |                                                                                                                  |
| Funktion:  | Holt Informationen über eine Diskette, Datei oder ein Verzeichnis von der Diskette. |                                                                                                                  |
| Parameter: | lock                                                                                | -> Zeiger auf die Lock-Structure der Datei oder des Verzeichnisses, über das Informationen geholt werden sollen. |
|            | info_data                                                                           | -> Zeiger auf eine leere Info_Data-Structure, in die die Informationen geschrieben werden sollen.                |
| Ergebnis:  | erfolg                                                                              | -> ist TRUE, wenn kein Fehler auftrat.                                                                           |
| Datentyp:  | struct Lock *lock;<br>struct Info_Data *info_data;<br>bool erfolg;                  |                                                                                                                  |

**Sonstiges:** Wenn Informationen über eine Diskette ermittelt werden sollen, enthält die `Info_Data`-Structure die Speicherkapazität, freie Blöcke und eventuelle Disketten-Fehler.

**Referenz:** Siehe auch CLI-Befehl `info`

### 10.1.10 Input

**Syntax:** `datei = Input();`

**Funktion:** Ermittelt den Zeiger auf die `FileHandle`-Structure mit der das Programm initialisiert wurde.

**Parameter:** Keine Parameter.

**Ergebnis:** `datei` -> Zeigerauf die `FileHandle`-Structure.

**Datentyp:** `struct FileHandle *datei;`

**Sonstiges:** Die oberste Eingabeebene ist im Normalfall die Boot-Diskette.

**Referenz:** Siehe auch `Output`

### 10.1.11 IOErr

**Syntax:** `fehler = IoErr();`

**Funktion:** Holt die Fehlernummer des zuletzt aufgetretenen IO-Fehlers.

**Parameter:** Keine Parameter.

**Ergebnis:** `fehler` -> Fehlernummer.

**Datentyp:** `int fehler;`

**Sonstiges:** Der Fehler kann anhand der Fehlernummer und dem Anhang D dieses Buches ermittelt werden.

**Referenz:** Siehe auch Anhang D

### 10.1.12 IsInteractive

**Syntax:** `bool = IsInteractive(datei);`

**Funktion:** Dieser Befehl stellt fest, ob die angegebene Datei einem virtuellen Terminal angehört.

**Parameter:** `datei` -> Zeiger auf das `FileHandle` der Datei, die überprüft werden soll.

|            |                                                                                                            |                                                                                              |
|------------|------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| Ergebnis:  | bool                                                                                                       | -> ist TRUE, wenn die Datei einem virtuelle Gerät angehört. Ansonsten ist bool gleich FALSE. |
| Datentyp:  | struct FileHandle *datei;                                                                                  |                                                                                              |
| Sonstiges: | Ein virtuelles Terminal ist zum Beispiel die Diskette. Nicht virtuell sind die Geräte NIL:, RAW: und CON:. |                                                                                              |

### 10.1.13 Lock

|            |                                                                                                                                                                                                                                                                                                           |                                                                                                                                                               |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | lock = Lock(name,mode);                                                                                                                                                                                                                                                                                   |                                                                                                                                                               |
| Funktion:  | Ermittelt die Lock-Structure einer Datei, bzw eines Verzeichnisses.                                                                                                                                                                                                                                       |                                                                                                                                                               |
| Parameter: | name                                                                                                                                                                                                                                                                                                      | -> Name der Datei, deren Lock-Structure-Zeiger ermittelt werden soll.                                                                                         |
|            | mode                                                                                                                                                                                                                                                                                                      | -> Für mode muß ACCESS_READ angegeben werden, um die Datei zum Lesen verwenden zu können. Wird ACCESS_WRITE angegeben, kann die Datei nur beschrieben werden. |
| Ergebnis:  | lock                                                                                                                                                                                                                                                                                                      | -> Zeiger auf die Lock-Structure der Datei, bzw. des Verzeichnisses.                                                                                          |
| Datentyp:  | char name[];<br>WORD mode;<br>struct Lock *lock;                                                                                                                                                                                                                                                          |                                                                                                                                                               |
| Sonstiges: | Im Gegensatz zu Open, wird bei Lock keine Datei geöffnet, sondern nur deren Lock-Structure zur Verfügung gestellt, was natürlich auch für Datei-Verzeichnisse gilt. So kann dieser Befehl verwendet werden, um zu überprüfen, ob eine bestimmte Datei vorhanden ist. Ist sie es nicht, ist lock gleich 0. |                                                                                                                                                               |
|            | Lock ist ein Puffer, in dem die wesentlichen Merkmale einer Datei festgehalten werden.                                                                                                                                                                                                                    |                                                                                                                                                               |

Da Dateien nicht gleichzeitig von mehreren Tasks aus beschrieben werden können, bedeutet `ACCESS_WRITE`, daß die Datei exklusiv ist. `ACCESS_READ` hingegen bedeutet, daß die Datei gleichzeitig von mehreren Tasks aus gelesen werden kann. Dazu muß allerdings der Befehl `DupLock` verwendet werden.

Referenz: Siehe auch `DupLock` und `UnLock`

### 10.1.14 Open

Syntax: `datei = Open(name,mode);`

Funktion: Öffnet eine Datei für Ein- oder Ausgabe.

Parameter: `name` -> Name der Datei, die geöffnet werden soll.  
`mode` -> `MODE_OLDFILE` öffnet eine bereits bestehende Datei mit dem Namen »name«.  
`MODE_NEWFILE` erstellt eine neue Datei mit dem Namen »name«.

Ergebnis: `datei` -> Zeiger auf die `FileHandle`-Structure der geöffneten Datei. »datei« ist gleich 0, wenn keine Datei geöffnet werden kann.

Datentyp: `char name[];`  
`WORD mode;`  
`struct FileHandle *datei;`

Sonstiges: Bevor eine bestehende Datei geöffnet wird, sollte zuvor mit `Lock` getestet werden, ob diese Datei überhaupt besteht.

Referenz: Siehe auch `Lock`, `Read`, `Write` und `Close`

### 10.1.15 Output

Syntax: `datei = Output();`

Funktion: Ermittelt den Zeiger auf die `FileHandle` der aktuellen Ausgabe-Ebene.

Parameter: Keine Parameter.

|            |                                                              |                                                   |
|------------|--------------------------------------------------------------|---------------------------------------------------|
| Ergebnis:  | datei                                                        | -> Zeiger auf die FileHandle-Structure der Ebene. |
| Datentyp:  | struct FileHandle *datei;                                    |                                                   |
| Sonstiges: | Die oberste Ausgabebene ist im Normalfall die Boot-Diskette. |                                                   |
| Referenz:  | Siehe auch Input                                             |                                                   |

### 10.1.16 ParentDir

|            |                                                                                                                                                                                          |                                                                                                                        |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | parlock = ParentDir(lock);                                                                                                                                                               |                                                                                                                        |
| Funktion:  | Ermittelt das Directory, in dem sich das angegebene Directory, bzw. die Datei befindet.                                                                                                  |                                                                                                                        |
| Parameter: | lock                                                                                                                                                                                     | -> Zeiger auf die Lock-Structure der Datei oder des Verzeichnisses, dessen Hauptverzeichnis ermittelt werden soll.     |
| Ergebnis:  | parlock                                                                                                                                                                                  | -> Zeiger auf die Lock-Structure des Verzeichnisses, in dem sich die angegebene Datei, bzw. das Verzeichniss befindet. |
| Datentyp:  | struct Lock *lock;<br>struct Lock *parlock;                                                                                                                                              |                                                                                                                        |
| Sonstiges: | Beispielsweise befindet sich die Datei »test« im Verzeichnis »demo«. Wird nun die Lock-Structure der Datei »test« angegeben, so erhält man die Lock-Structure des Verzeichnisses »demo«. |                                                                                                                        |
|            | Wenn keine Datei »test« gefunden wird, ist parlock gleich 0.                                                                                                                             |                                                                                                                        |

### 10.1.17 Read

|            |                                         |                                                                                                 |
|------------|-----------------------------------------|-------------------------------------------------------------------------------------------------|
| Syntax:    | echt = Read(datei,ziel,länge);          |                                                                                                 |
| Funktion:  | Liest Daten aus einer geöffneten Datei. |                                                                                                 |
| Parameter: | datei                                   | -> Zeiger auf die FileHandle-Structure der Datei, aus der gelesen werden soll.                  |
|            | ziel                                    | -> Zeiger auf das erste Byte des Speicherbereiches, in den die Daten geschrieben werden sollen. |
|            | länge                                   | -> Länge des Speicherbereiches in Bytes.                                                        |

|            |                                                                                                                                                                                                                                                                       |                                                  |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|
| Ergebnis:  | echt                                                                                                                                                                                                                                                                  | -> Anzahl der Bytes, die gelesen werden konnten. |
| Datentyp:  | struct FileHandle *datei;<br>ULONG ziel;<br>int länge;<br>int echt;                                                                                                                                                                                                   |                                                  |
| Sonstiges: | Aus einer Datei kann nur dann gelesen werden, wenn sie zuvor mit Open geöffnet wurde. Durch Open erhält man auch den Zeiger auf die File-Handle-Structure der Datei.<br><br>»echt« ist maximal so lang wie »länge«. Ist »echt« gleich -1, ist ein Fehler aufgetreten. |                                                  |
| Referenz:  | Siehe auch Write, Open und Close                                                                                                                                                                                                                                      |                                                  |

### 10.1.18 Rename

|            |                                                                                                                   |                                                             |
|------------|-------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|
| Syntax:    | erfolg = Rename(alt,neu);                                                                                         |                                                             |
| Funktion:  | Ändert Datei- oder Dateiverzeichnisnamen.                                                                         |                                                             |
| Parameter: | alt                                                                                                               | -> Alter Name der Datei oder des Verzeichnisses.            |
|            | neu                                                                                                               | -> Neuer Name der Datei oder des Verzeichnisses.            |
| Ergebnis:  | erfolg                                                                                                            | -> TRUE, wenn kein Fehler aufgetreten ist. Ansonsten FALSE. |
| Datentyp:  | char alt[], neu[];<br>bool erfolg;                                                                                |                                                             |
| Sonstiges: | Wenn schon eine Datei oder ein Verzeichnis mit dem Namen "neu" existiert, wird der Name von "alt" nicht geändert. |                                                             |
| Referenz:  | Siehe auch CLI-Befehl rename                                                                                      |                                                             |

## 10.1.19 Seek

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                       |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | altpos = Seek(datei,pos,mode);                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                       |
| Funktion:  | Stellt den »Schreib/Lesezeiger« einer Datei auf eine neue Position.                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                                       |
| Parameter: | datei                                                                                                                                                                                                                                                                                                                                                                                                                                 | -> Zeiger auf die FileHandle-Structure der Datei, in der der "Schreib/Lesezeiger" neu gesetzt werden soll.                                                                            |
|            | pos                                                                                                                                                                                                                                                                                                                                                                                                                                   | -> Neue Position des Zeigers.                                                                                                                                                         |
|            | mode                                                                                                                                                                                                                                                                                                                                                                                                                                  | -> kann sein:<br>OFFSET_BEGINNING: neue Position vom Dateianfang aus<br>OFFSET_CURRENT: neue Position von der momentanen Position aus<br>OFFSET_END: neue Position vom Dateiende aus. |
| Ergebnis:  | altpos                                                                                                                                                                                                                                                                                                                                                                                                                                | -> Alte Position des Zeigers.                                                                                                                                                         |
| Datentyp:  | struct FileHandle *datei;<br>int pos;<br>WORD mode;<br>int altpos;                                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                       |
| Sonstiges: | Nach dem Öffnen einer Datei mit Open, steht der Zeiger am Anfang der Datei. Mit Read oder Write rückt der Zeiger in der Datei entsprechend der Anzahl der gelesenen oder geschriebenen Bytes weiter. Mit Seek kann dieser Zeiger nun vom Programmierer aus selbstständig gesteuert werden. Wählt man für "mode" OFFSET_END und für pos gleich -35, so bedeutet dies, daß der Zeiger 35 Bytes vor dem Ende der Datei positioniert ist. |                                                                                                                                                                                       |

### 10.1.20 SetComment

|            |                                                                                  |                                                                                      |
|------------|----------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| Syntax:    | erfolg = SetComment(name,kommentar);                                             |                                                                                      |
| Funktion:  | Schreibt einen dateispezifischen Kommentar in das Dateiverzeichnis der Diskette. |                                                                                      |
| Parameter: | name                                                                             | -> Name der Datei oder des Verzeichnisses, für das ein Kommentar gesetzt werden soll |
|            | kommentar                                                                        | -> Kommentartext, der maximal 80 Zeichen lang sein darf.                             |
| Ergebnis:  | erfolg                                                                           | -> ist TRUE, wenn kein Fehler auftrat.                                               |
| Datentyp:  | char name[], kommentar[];<br>bool erfolg;                                        |                                                                                      |
| Sonstiges: | Der Kommentar erscheint beim Auflisten des Directory.                            |                                                                                      |

### 10.1.21 SetProtection

|            |                                                                                                                                                                                                                                                                                                             |                                                                       |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|
| Syntax:    | erfolg = SetProtection(name,mode);                                                                                                                                                                                                                                                                          |                                                                       |
| Funktion:  | Schützt eine Datei oder ein Dateiverzeichnis.                                                                                                                                                                                                                                                               |                                                                       |
| Parameter: | name                                                                                                                                                                                                                                                                                                        | -> Name der Datei oder des Verzeichnisses, das geschützt werden soll. |
|            | mode                                                                                                                                                                                                                                                                                                        | -> Siehe Sonstiges.                                                   |
| Ergebnis:  | erfolg                                                                                                                                                                                                                                                                                                      | -> ist TRUE, wenn kein Fehler auftrat.                                |
| Datentyp:  | char name[];<br>LONG mode;<br>bool erfolg;                                                                                                                                                                                                                                                                  |                                                                       |
| Sonstiges: | »mode« ist vom Typ LONG, also 32 Bit lang. Die Bits 31 bis 4 haben derzeit noch keine Bedeutung.<br>Bit 3 setzt den Leseschutz (nicht lesbar).<br>Bit 2 setzt den Schreibschutz (nicht überschreibbar)<br>Bit 1 setzt den Ausführschutz (nicht ausführbar).<br>Bit 0 setzt den Löserschutz (nicht löschar). |                                                                       |



Datentyp: struct FileHandle \*datei;  
LONG zeit;  
bool bool;

Sonstiges: Die Zeitspanne, innerhalb der das Zeichen bereitgestellt werden muß, ist in Mikrosekunden, also millionstel Sekunden, anzugeben.

### 10.1.24 Write

Syntax: `erfolg = Write(datei,daten,länge);`

Funktion: Schreibt Daten in eine geöffnete Datei.

Parameter: `datei` -> Zeiger auf die FileHandle-Structure der Datei, in die geschrieben werden soll.

`daten` -> Zeiger auf das erste Byte des Speicherbereiches, der in die Datei geschrieben werden soll.

`länge` -> Länge des Speicherbereiches in Bytes.

Ergebnis: `erfolg` -> ist -1, wenn ein Fehler aufgetreten ist (beispielsweise "Disk Full") Ist `erfolg` größer als 0, ist kein Fehler aufgetreten

Datentyp: struct FileHandle \*datei;  
ULONG daten;  
int länge;  
int erfolg;

Sonstiges: In eine Datei kann nur dann geschrieben werden, wenn sie zuvor mit Open geöffnet wurde. Durch Open erhält man auch den Zeiger auf die File-Handle-Structure der Datei.

Referenz: Siehe auch Read, Open und Close

## 10.2 DOS-Demonstration

Diese Demonstration führen wir als eigenes Unter-Kapitel auf, da sie aus zwei Teilen besteht. Der erste Teil ist ein Basic-Programm, das Bilder in ein Format wandelt, das mit dem zweiten Programm eingelesen werden kann.

Um Bilder einlesen zu können, muß also zuerst das Basic-Programm gestartet werden (es befindet sich auch auf der mitgelieferten Diskette). Dazu muß sich allerdings der Amiga-Basic-Interpreter auf der Diskette befinden. Anschließend fragt es nach dem Namen des Bildes, das konvertiert werden soll. Dieses Bild muß im Graphicraft-Format abgespeichert sein, nicht im DeLuxePaint-Format. Soll ein DeluxePaint-Bild konvertiert werden, so muß dieses zuvor mit Graphicraft geladen und wieder gespeichert worden sein.

Anschließend fragt das Basic-Programm nach dem Namen, unter dem das konvertierte Bild gespeichert werden soll. Das Programm benötigt dann einige Zeit, um das Bild zu konvertieren.

Da das Basic-Programm sehr einfach ist, speichert es nur die Bilddaten ab, nicht die Farbinformationen. Diese müssen später im eigenen C-Programm gesetzt werden. Es ist allerdings möglich, die Farben mitabzuspeichern, was Sie selbst ändern können.

In Ihr C-Programm können Sie dann die C-Routine einbinden, die das Bild einliest. Soll ein Bild nicht eingelesen, sondern auf Diskette gespeichert werden, so sind die »Read«-Befehle gegen »Write«-Befehle auszutauschen.

Es besteht noch eine weitere Einschränkung. Beide Routinen laden nur Bilder mit der Auflösung 320 mal 200 Pixel. Aber auch dies kann von Ihnen leicht verändert werden.

Aber nun zu den Demonstrationen. Auf der Diskette befindet sich ein Bild, das eingeladen wird, wenn die DOS-Demonstration gestartet wird.

```

REM *****
REM Konvertierungsprogramm von IFF in JKFK
REM *****

REM Dieses Programm konvertiert ein Graphicraftbild in ein eigenes Format
REM Dieses Format besteht allerdings nur aus den Bildinformationen
REM der einzelnen Bitplanes, die hintereinander abgespeichert werden.

REM *****

CLS
INPUT "IFF - File :";a$
PRINT
INPUT "JKFK - File :";b$

REM Screen und Window oeffnen
SCREEN 1,320,200,5,1
WINDOW 1
WINDOW 2,,(0,11)-(310,185),0,1
WINDOW OUTPUT 2

REM Datenfelder dimensionieren
DIM ad$(4)
DIM fr(31),fg(31),fb(31)

REM IFF-File lesen
OPEN a$ FOR INPUT AS #1
a$= INPUT$(8,#1)
a$= INPUT$(4,#1)
IF a$<>"ILBM" THEN fehler
bmhd:
a$=INPUT$(4,#1)
a&=CVL(INPUT$(4,#1))
IF a$="CMAP" THEN GOTO cmap
IF a$="BODY" THEN GOTO body
a$=INPUT$(a&,#1)
GOTO bmhd

body:
REM Plane-Startadressen
FOR i=0 TO 4
ad$(i)=PEEK(PEEK(WINDOW(8)+4)+8+4*i)
NEXT i

REM IFF-Bilddaten lesen
FOR y=0 TO 199
FOR i=0 TO 4
ba&=ad$(i)+40*y
FOR x=0 TO 36 STEP 4
POKEL ba&+x,CVL(INPUT$(4,#1))
NEXT x,i,y
CLOSE #1

REM JKFK - File abspeichern
OPEN b$ FOR OUTPUT AS #1 LEN=12500
a=0
block:
FOR y= 0 TO 199
ba& =ad$(a)+40*y
FOR x=0 TO 36 STEP 4
PRINT #1,MKL$(PEEK(ba&+x));
NEXT x,y

```

```

IF a<4 THEN count: ELSE CLOSE #1
END
count:
a=a+1
GOTO block:

REM Farbmappe lesen
cmap:
FOR i=0 TO a&/3-1
 fr(i)=ASC(INPUT$(1,#1))/16
 fg(i)=ASC(INPUT$(1,#1))/16
 fb(i)=ASC(INPUT$(1,#1))/16
 PALETTE i,fr(i)/16,fg(i)/16,fb(i)/16
NEXT i
GOTO bmhd

1 /*****
2
3 DOS-Demonstration
4 last update 26/05/87
5 von Frank Kremser und Joerg Koch
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Diese Demonstration oeffnet eine Datei mit dem Namen `Titelbild` und
11 laedt diese in den geoeffneten Screen, dabei wird ein eigenes Screen-
12 Format verwendet. Bilder die Graphicraft erstellt wurden, muessen
13 zunaechst mit dem Basic-Konvertierer konvertiert werden, bevor sie
14 mit dieser Routine ladbar sind. DPaint Bilder muessen dagan zunaechst
15 mit Graphicraft geladen, abgespeichert und dann konvertiert werden !
16
17 *****/
18
19 #include <exec/types.h> /* Include-Files laden */
20 #include <exec/tasks.h>
21 #include <exec/libraries.h>
22 #include <exec/devices.h>
23 #include <devices/keymap.h>
24 #include <graphics/copper.h>
25 #include <graphics/display.h>
26 #include <graphics/gfxbase.h>
27 #include <graphics/text.h>
28 #include <graphics/view.h>
29 #include <graphics/gels.h>
30 #include <graphics/regions.h>
31 #include <graphics/sprite.h>
32 #include <hardware/blit.h>
33 #include <libraries/dos.h>
34 #include <intuition/intuition.h>
35 #include <intuition/intuitionbase.h>
36
37
38 struct IntuitionBase *IntuitionBase; /* Lib-Zeiger */
39 struct GfxBase *GfxBase;
40 ULONG DosBase;
41
42 UBYTE *puffer[5]; /* max 5 Puffer-Zeiger */
43
44 struct Screen *screen;
45
46 struct NewScreen ns = /* Screen definieren */

```

```

47 {
48 0,
49 0,
50 320,
51 200,
52 5,
53 0,
54 1,
55 0,
56 CUSTOMSCREEN,
57 NULL,
58 NULL,
59 NULL,
60 NULL
61 };
62
63
64 main()
65 {
66 LONG warte;
67 ULONG datei;
68 struct RastPort *rp;
69 struct BitMap *ptr;
70
71 if ((GfxBase = (struct GfxBase *)
72 OpenLibrary("graphics.library", 0)) == 0) exit();
73
74 if ((IntuitionBase = (struct IntuitionBase *)
75 OpenLibrary("intuition.library", 0)) == 0) exit();
76 /* Libs oeffnen */
77 if ((DosBase =
78 OpenLibrary("dos.library", 0)) == 0) exit();
79 /* Screen oeffnen */
80 if ((screen = (struct Screen*) OpenScreen(&ns)) == NULL) exit();
81
82 for(warte = 0; warte < 31; ++warte)
83 SetRGB4(&screen->ViewPort,warte,0,0,0); /* Farben auf Schwarz setzen */
84
85 rp = &screen->RastPort;
86 ptr = rp->BitMap;
87 puffer[0] = ptr->Planes[0]; /* Zeiger auf BitPlanes ermitteln */
88 puffer[1] = ptr->Planes[1];
89 puffer[2] = ptr->Planes[2];
90 puffer[3] = ptr->Planes[3];
91 puffer[4] = ptr->Planes[4];
92
93 datei = Open("titelbild",MODE_OLDFILE); /* Datei oeffnen */
94 if(datei == 0)
95 {
96 printf("Keinen File geoeffnet !!!\n"); /* Fehler */
97 CloseScreen(screen);
98 exit();
99 };
100
101 Read(datei,puffer[0],8000); /* Datei einladen */
102 Read(datei,puffer[1],8000); /* Die Laenge berechnet sich aus der */
103 Read(datei,puffer[2],8000); /* Groesse einer Bitplane = */
104 Read(datei,puffer[3],8000); /* 320 x 200 / 8 */
105 Read(datei,puffer[4],8000);
106
107 /* Soll kein Bild geladen, sondern */
108 /* gespeichert werden, muss anstatt */
109 /* 'Read' 'Write' verwendet werden */
110 Close(datei);

```

```
110
111 SetRGB4(&screen->ViewPort,0,0,0,0); /* Farben fuer Titelbild */
112 SetRGB4(&screen->ViewPort,1,15,15,15);
113 SetRGB4(&screen->ViewPort,2,14,0,0);
114 SetRGB4(&screen->ViewPort,3,14,1,0);
115 SetRGB4(&screen->ViewPort,4,14,2,0);
116 SetRGB4(&screen->ViewPort,5,14,4,0);
117 SetRGB4(&screen->ViewPort,6,14,5,0);
118 SetRGB4(&screen->ViewPort,7,14,6,0);
119 SetRGB4(&screen->ViewPort,8,15,8,0);
120 SetRGB4(&screen->ViewPort,9,15,9,0);
121 SetRGB4(&screen->ViewPort,10,15,11,0);
122 SetRGB4(&screen->ViewPort,11,15,12,0);
123 SetRGB4(&screen->ViewPort,12,15,14,0);
124 SetRGB4(&screen->ViewPort,13,15,15,0);
125 SetRGB4(&screen->ViewPort,14,15,15,4);
126 SetRGB4(&screen->ViewPort,15,13,15,4);
127 SetRGB4(&screen->ViewPort,16,10,15,3);
128 SetRGB4(&screen->ViewPort,17,7,15,3);
129 SetRGB4(&screen->ViewPort,18,3,15,2);
130 SetRGB4(&screen->ViewPort,19,2,15,4);
131 SetRGB4(&screen->ViewPort,20,1,15,6);
132 SetRGB4(&screen->ViewPort,21,0,15,0);
133 SetRGB4(&screen->ViewPort,22,1,15,13);
134 SetRGB4(&screen->ViewPort,23,0,14,15);
135 SetRGB4(&screen->ViewPort,24,0,12,15);
136 SetRGB4(&screen->ViewPort,25,0,10,15);
137 SetRGB4(&screen->ViewPort,26,0,9,15);
138 SetRGB4(&screen->ViewPort,27,0,7,15);
139 SetRGB4(&screen->ViewPort,28,0,5,15);
140 SetRGB4(&screen->ViewPort,29,0,3,15);
141 SetRGB4(&screen->ViewPort,30,0,2,15);
142 SetRGB4(&screen->ViewPort,31,0,0,15);
143
144 for(warte = 0; warte < 750000; ++warte);
145
146 CloseScreen(screen); /* Libs und Screen schliessen */
147 CloseLibrary(DosBase);
148 CloseLibrary(IntuitionBase);
149 CloseLibrary(GfxBase);
150 }
```



## Der Drucker

Eines der wichtigsten Geräte des Amiga ist der Drucker. Ohne Drucker geht heutzutage gar nichts mehr. Selbst der größte Teil dieses Buches wäre ohne die Verwendung eines Druckers nicht zustande gekommen. Deshalb wollen wir diesen Teil der Hardware nicht unter den Tisch fallen lassen.

Der Amiga besitzt vier Möglichkeiten, einen Drucker ansprechen zu können:

**PRT:** (Amiga-DOS Printer.device):

Dies haben Sie sicherlich schon einmal kennengelernt, wenn Sie im CLI Dateien auf dem Drucker ausgeben wollten. Unter CLI ist die Anwendung sehr einfach, denn es wird einfach nur die Geräteausgabe umgeleitet. Einen Text unter CLI können Sie mit »TYPE > PRT: Dateixyz« ausdrucken.

**SER:** (Amiga-DOS Seriell.device):

Wenn der Drucker am Serial-Port angeschlossen ist, kann er mit SER: angesprochen werden.

**PAR:** (Amiga-DOS Parallel.device):

Befindet sich der Drucker aber am Parallel-Port des Amiga, so muß er mit der PAR: angesprochen werden.

Die letzte und variabelste Möglichkeit ist das Ansprechen der printer.device, die wir im Kapitel 1.2 »Die Devices« schon kennengelernt haben.

Printer.device ist eine Art »Software-Drucker-Interface«, die unter Verwendung des mit Preferences eingestellten Printer-Typs, die ankommenden Daten für diesen übersetzt. Printer.device stellt dafür drei verschiedene Structures zur Verfügung. Der erste, IOPrtCmdReq, dient zur Übertragung von Steuerzeichen, die den Drucker z.B. von »Italic« auf »Normal« umstellen können. Eine weitere Structure, die IODRPREq, wird benötigt, wenn der Inhalt des Screens auf dem Drucker ausgegeben werden soll. Die letzte, ebenfalls wichtige Structure, stellt die printer.device nicht selbst zur Verfügung. Sie muß von Exec geladen werden und dient zum Ausdrucken von Texten.

## 11.1 Druckerausgabe über Amiga-DOS

»PRT:« ist eine der einfachsten Möglichkeiten, Daten zum Drucker zu senden. Der Nachteil ist aber, daß man ausschließlich Texte senden kann.

Dies geschieht ganz einfach indem man mit

```
datei = Open("PRT:", MODE_NEWFILE);
```

den Druckerport öffnet und anschließend mit

```
echt = Write(datei, &text, sizeof(text));
```

den gewünschten Text ausgibt. Die Variable »echt« entspricht einer Integervariable, die die Nummer des Fehlers enthält, falls einer aufgetreten ist. »&text« entspricht dem Zeiger auf den Text. Der letzte Parameter ist die Länge des Textes.

Nach diesem Befehl muß der Druckerport wieder geschlossen werden, da sonst die Ausgabe des CLI über den Drucker läuft. Dies geschieht mit dem Befehl

```
Close(datei);
```

»PAR:« und »SER:« können ebenfalls angewendet werden. Dabei ist aber zu beachten, daß Amiga-DOS die Daten nicht aufbereitet, sondern direkt sendet.

## 11.2 Die Printer.device

Die printer.device ist die Software-Schnittstelle zum Drucker. Mit ihr können Steuermodi gesetzt, sowie Screens- und Texte gedruckt werden.

Zunächst muß man, wie im Kapitel Devices schon beschrieben, einen Port und die jeweilige Device, in diesem Fall die printer.device, öffnen. Danach kann durch Übergabe einer Structure die gewünschte Funktion »eingeschaltet« werden. Es stehen drei Structures und somit drei verschiedene Funktionen zur Verfügung:

- Screen-Hardcopy
- Übergeben von Steuerzeichen
- Ausdruck von Texten

Die Screen-Hardcopy-Structure IODRPREq ist für das Ausdrucken eines Screens, bzw. eines RastPorts zuständig. Sie ist wie folgt festgelegt :

```
struct IODRPREq
{
 struct Message io_Message;
 struct Device *io_Device;
 struct Unit *io_Unit;
 UWORD io_Command;
 UBYTE io_Flags;
 BYTE io_Error;
 struct RastPort *io_RastPort;
 struct ColorMap *io_ColorMap;
 ULONG io_Modes;
 UWORD io_SrcX;
 UWORD io_SrcY;
 UWORD io_SrcWidth;
 UWORD io_SrcHeight;
 LONG io_DestCols;
 LONG io_DestRows;
 UWORD io_Special;
};
```

|             |                                           |
|-------------|-------------------------------------------|
| io_Message  | Nachrichten-Struktur.                     |
| io_Device   | Zeiger auf die Device.                    |
| io_Unit     | Zeiger auf den eigenen Printer-Driver.    |
| io_Command  | Device-Kommando.                          |
| io_Flags    | Ein/Ausgabe-Flags.                        |
| io_Error    | Fehler-Nummer.                            |
| io_RastPort | Der RastPort, der ausgegeben werden soll. |
| io_ColorMap | Farb-Mappe.                               |
| io_Modes    | Der ViewMode.                             |
| io_SrcX     | x Ursprung des RastPorts.                 |
| io_SrcY     | y Ursprung des RastPorts.                 |

|                           |                                                                               |
|---------------------------|-------------------------------------------------------------------------------|
| <code>io_SrcWidth</code>  | Breite des RastPorts.                                                         |
| <code>io_SrcHeight</code> | Höhe des RastPorts.                                                           |
| <code>io_DestCols</code>  | Breite des Ausdrucks in Punkten.                                              |
| <code>io_DestRows</code>  | Höhe des Ausdrucks in Punkten.                                                |
| <code>io_Special</code>   | Hier können verschiedene Flags gesetzt werden, die den Ausdruck beeinflussen: |

| definition:                    | Hex-Wert: | Erklärung:                                                                                                                                        |
|--------------------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Special_milcols</code>   | 0x001:    | Bestimmung der Höhe des Dumps in 1/1000.                                                                                                          |
| <code>Special_milrows</code>   | 0x002:    | Bestimmung der Breite des Dumps in 1/1000.                                                                                                        |
| <code>Special_fullcols</code>  | 0x004:    | Der Ausdruck soll in maximaler Breite geschehen. <code>io_DestCols</code> und <code>io_DestRows</code> werden ignoriert.                          |
| <code>Special_fullrows</code>  | 0x008:    | Der Ausdruck soll in maximaler Höhe geschehen. <code>io_DestCols</code> und <code>io_DestRows</code> werden ignoriert.                            |
| <code>Special_fraccols</code>  | 0x010:    | Die Höhe ist ein Bruchteil von FULLCOLS.                                                                                                          |
| <code>SpecialL_fracrows</code> | 0x020:    | Die Breite ist ein Bruchteil von FULLROWS.                                                                                                        |
| <code>Sprcial_aspect</code>    | 0x080:    | <code>io_DestRows</code> wird ignoriert und in Abhängigkeit von <code>io_DestCols</code> berechnet, so daß die Seitenverhältnisse übereinstimmen. |
| <code>Special_density1</code>  | 0x100:    | Dichte-Bits, geringste Auflösung.                                                                                                                 |
| <code>Special_density2</code>  | 0x200:    | Nächst höhere.                                                                                                                                    |
| <code>Special_density3</code>  | 0x300:    | Nächst höhere.                                                                                                                                    |
| <code>Special_density4</code>  | 0x400:    | Höchste Auflösung.                                                                                                                                |

Bei der Anwendung dieser Structure brauchen nicht alle Variablen belegt zu werden, da ein Teil der Deklaration von `OpenPort()`, sowie `OpenDevice()`

übernommen wird. Eine praktische Anwendung dieser Structure zeigt die Demonstration »DumpScreen«, die Sie im Anschluß an dieses Kapitel finden.

Die zweite Structure, die in der printer.device enthalten ist, ist die IOPrtCmdReq-Structure. Sie wird zum Übergeben von Steuer-Zeichen an den Drucker verwendet.

```
struct IOPrtCmdReq
{
 struct Message io_Message;
 struct Device *io_Device;
 struct Unit *io_Unit;
 UWORD io_Command;
 UBYTE io_Flags;
 BYTE io_Error;
 UWORD io_PrtCommand;
 UBYTE io_Parm0;
 UBYTE io_Parm1;
 UBYTE io_Parm2;
 UBYTE io_Parm3;
};
```

|               |                                                                                                         |
|---------------|---------------------------------------------------------------------------------------------------------|
| io_Message    | Nachrichten-Struktur.                                                                                   |
| io_Device     | Zeiger auf die Device.                                                                                  |
| io_Unit       | Zeiger auf den eigenen Printer-Driver.                                                                  |
| io_Command    | Device - Kommando.                                                                                      |
| io_Flags      | Ein/Ausgabe - Flags.                                                                                    |
| io_Error      | Fehler-Nummer.                                                                                          |
| io_PrtCommand | Hier muß das Kommando für den Drucker eingetragen werden. Die Drucker-Kommandos finden Sie im Anhang F. |
| io_Parm0      | Parameter für das Kommando.                                                                             |
| io_Parm1      | Parameter für das Kommando.                                                                             |
| io_Parm2      | Parameter für das Kommando.                                                                             |
| io_Parm3      | Parameter für das Kommando (siehe Anhang F).                                                            |

Nachdem Sie mit Hilfe dieser Structure den Druckmodus eingestellt haben, kann der Text ausgedruckt werden. Ein Beispiel hierfür finden Sie in der »Dosprinterdemo«.

Der Text, der nach dem Einstellen eines Drucker-Kommandos gedruckt werden soll, muß mit einer weiteren Structure übergeben werden. Diese Structure ist die allgemeine I/O-Structure von EXEC. Nicht alles von dieser Structure muß deklariert sein, so daß wir hier nun den benötigten Teil vorstellen:

```
struct IOStdReq
{
 struct Message io_Message;
```

```
APTR io_Data;
ULONG io_Lenght;
}
```

**io\_Message** : Nachrichten - Struktur.

**io\_Data** : Zeiger auf den Text, der ausgegeben werden soll.

**io\_Lenght** : Länge des Textes.

Wird in **Lenght -1** eingetragen, so bedeutet dies, daß der Text mit 0 enden muß.

```
1 /*****
2
3 DOS-Printer-Demonstration
4 last update 26/05/1987
5 von Frank Kremser und Joerg Koch
6 (c) Markt & Technik 1987
7 *****/
8
9 Diese Demonstration verdeutlicht, wie von DOS aus auf den Drucker
10 zugegriffen werden kann. DOS-Printer-Demo kann nur vom CLI aus
11 gestartet werden.
12
13 *****/
14
15 #include "exec/types.h" /* Include-Files laden */
16 #include "exec/exec.h"
17 #include "libraries/dos.h"
18 #include "intuition/intuition.h"
19
20 ULONG DosBase; /* Lib-Zeiger */
21 ULONG datei; /* Zeiger auf datei */
22
23 char text[] = "Drucker-Demonstration\n"; /* Auszugebender Text */
24
25 main()
26 {
27 int echt; /* int Variable fuer Fehlermeldung bei Write() */
28
29 DosBase = OpenLibrary("dos.library",0); /* DOS-Bibliothek oeffnen */
30 if(DosBase == NULL) exit();
31
32 datei = Open("PRT:",MODE_NEWFILE); /* Druckerkanal oeffnen */
33
34 echt = Write(datei,&text[0],sizeof(text)); /* Text schreiben */
35
36 Close(datei); /* Druckerkanal schliessen */
37
38 CloseLibrary(DosBase); /* DOS-Bibliothek schliessen */
39 }
```

```

1 /*****
2
3 Druckerdemonstration
4 last update 26/05/87
5 von Joerg Koch und Frank Kremser
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Diese Demonstration verdeutlicht den Umgang mit der Printer-Device.
11 Es wird gezeigt, wie Druckereinstellungen geaendert werden koennen
12 und wie Text ausgegeben werden kann:
13 Es wird ein Text in Italics ausgegeben. (Auf einem Epson-Drucker
14 getestet.
15
16 *****/
17
18
19 #include <exec/types.h> /* Include-Files einlesen */
20 #include <exec/exec.h>
21 #include <intuition/intuition.h>
22 #include <devices/printer.h>
23
24 struct MsgPort *CreatePort(); /* Zuweisen der Message-Ports */
25 struct MsgPort *printerPort;
26
27 struct IOStdReq druckrequest; /* Structure fuer drucken */
28 struct IOPrctCmdReq prefrequest; /* Structure fuer einstellen */
29
30 char text[] = "ABCDEFGH-IJKLMNOPQRSTUVWXYZ\n\0"; /* Auszugebender Text */
31
32
33 main()
34 {
35 int fehler;
36
37 printerPort = (struct MsgPort *)CreatePort("printer.port",0);
38 /* Drucker-Device oeffnen */
39 fehler = OpenDevice("printer.device",0,&prefrequest,0);
40 if(fehler != 0)
41 {
42 DeletePort(printerPort); /* Wenn Fehler, Port loeschen */
43 exit();
44 };
45
46 /* Antwort-Port = Printer-Port */
47 prefrequest.io_Message.mn_ReplyPort = printerPort;
48 prefrequest.io_Command = PRD_PRTCOMMAND; /* Print - Kommando */
49 prefrequest.io_Prm0 = 0; /* Italics ein */
50 prefrequest.io_Prm1 = 0; /* keine Parameter */
51 prefrequest.io_Prm2 = 0;
52 prefrequest.io_Prm3 = 0;
53
54 fehler = DoIO(&prefrequest); /* Werte uebergeben */
55
56 CloseDevice(&prefrequest); /* Pref-Device schliessen */
57 /* Druck-Device oeffnen */
58 fehler = OpenDevice("printer.device",0,&druckrequest,0);
59 if(fehler != 0)
60 {
61 DeletePort(printerPort);
62 exit();
63 };
64

```

```
65 druckrequest.io_Message.mn_ReplyPort = printerPort;
66 druckrequest.io_Command = CMD_WRITE; /* Kommando Schreiben */
67 druckrequest.io_Data = (APTR)&text[0]; /* Text uebergeben */
68 druckrequest.io_Length = -1; /* Text wird von >0< beendet */
69
70 fehler = DoIO(&druckrequest); /* Daten uebergeben */
71
72 CloseDevice(&druckrequest); /* Drucker-Device schliessen */
73 DeletePort(printerPort); /* Drucker-Port schliessen */
74 }

1 /*****
2
3 Dump-Demonstration
4 last update 26/05/87
5 von Frank Kremser und Joerg Koch
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Diese Demonstration gibt den Inhalt der Workbench auf dem Drucker aus
11
12 *****/
13
14 #include <exec/types.h> /* Include-Files laden */
15 #include <exec/exec.h>
16 #include <intuition/intuition.h>
17 #include <intuition/intuitionbase.h>
18 #include <devices/printer.h>
19
20 extern struct MsgPort *CreatePort(); /* Message-Port zuweisen */
21
22 struct IntuitionBase *IntuitionBase;
23 struct MsgPort *printerPort;
24 struct IOGRPReq request; /* Dump-Structure */
25 struct Window *window;
26
27 struct NewWindow nw = /* Windowdefinition */
28 {
29 0,
30 246,
31 640,
32 10,
33 1,
34 2,
35 0,
36 WINDOWDRAG:ACTIVATE!WINDOWDEPTH,
37 NULL,
38 NULL,
39 "Hardcopyroutine von Joerg Koch und Frank Kremser vom 19.4.87 um 4:46",
40 NULL,
41 NULL,
42 0,
43 0,
44 0,
45 0,
46 WBENCHSCREEN
47 };
48
49
50 main()
```

```

51 {
52 int fehler;
53 struct Screen *screen;
54
55 IntuitionBase = (struct IntuitionBase *) /* Intuition oeffnen */
56 OpenLibrary("intuition.library",0);
57 if(IntuitionBase == NULL) exit();
58
59 window = (struct Window *)OpenWindow(&nw); /* Window oeffnen */
60 if(window == NULL)
61 {
62 CloseLibrary(IntuitionBase);
63 exit();
64 };
65
66 screen = window->WScreen; /* Screen-Structure der */
67 /* Workbench ermitteln */
68 printerPort = (struct MsgPort *)CreatePort("printer.port",0);
69 if(printerPort == NULL) exit(); /* Drucker-Port oeffnen */
70
71 fehler = OpenDevice("printer.device",0,&request,0);
72 if(fehler != 0) /* Drucker-Device oeffnen */
73 {
74 CloseLibrary(IntuitionBase);
75 DeletePort(printerPort);
76 exit();
77 };
78 /* Antwort-Port zuweisen */
79 request.io_Message.mn_ReplyPort = printerPort;
80 request.io_Command = PRD_DUMPRT;
81 request.io_RastPort = &screen->RastPort; /* RastPort */
82 request.io_ColorMap= screen->ViewPort.ColorMap; /* Farbtabelle */
83 request.io_Modes = screen->ViewPort.Modes; /* Mode */
84 request.io_SrcX = 0; /* des auszugebenden Screens */
85 request.io_SrcY = 0;
86 request.io_SrcWidth = screen->Width;
87 request.io_SrcHeight = screen->Height;
88 request.io_DestCols = 960; /* Anzahl der Spalten auf dem Papier */
89 request.io_DestRows = 0;
90 request.io_Special = 0x00B0; /* ASPECT */
91 /* SrcX und SrcY geben die Position an, ab der gedruckt werden soll.
92 SrcWidth und ScrHeight geben die Hoehe des zu druckenden Bereiches an. Mit
93 DestCols wird die Anzahl der Punkte gesetzt, die das Bild auf dem Drucker
94 besitzen soll. DestRows braucht nicht gesetzt zu werden, da in Special
95 ASPECT = 0x00B0 gesetzt ist, was die Reihen automatisch in das richtige
96 Verhaeltniss zu den Spalten bringt */
97
98 fehler = DoIO(&request); /* Ausgabe auf dem Drucker */
99
100 CloseDevice(&request); /* Printer-Device schliessen */
101 DeletePort(printerPort); /* Printer-Port schliessen */
102 CloseWindow(window); /* Window schliessen */
103 }

```



## Die Workbench

Die Workbench ist wohl jedem Amiga-Benutzer als graphische Arbeitsoberfläche bekannt. Doch daß sie auch einige eigene Befehle besitzt, dürfte vielen ein Geheimnis sein. Bei genauerer Betrachtung ist es jedoch einleuchtend, daß dem so sein muß, denn wie sonst könnte ein Icon von der Diskette geholt werden oder auch wieder darauf abgespeichert werden, wenn es verändert wurde.

Die Workbench-Befehle sind normalerweise nicht sehr interessant, da sie wohl äußerst selten Verwendung finden. Es ist aber denkbar, daß ein Programm die Steuerung der Workbench teilweise übernimmt, wofür die folgenden Befehle vonnöten sind.

## 12.1 AddFreeList

- Syntax:** fehler = AddFreeList(freelist, anfang, länge);
- Funktion:** Fügt einen Speicherbereich in eine FreeList-Structure ein.
- Parameter:**
- freelist -> Zeiger auf die FreeList-Structure, in die der Speicherbereich eingefügt werden soll.
  - anfang -> Zeiger auf das erste Byte des Speicherbereiches, der eingefügt werden soll.
  - länge -> Länge des Speicherbereiches in Bytes.
- Ergebnis:** fehler -> ist 0, wenn kein Fehler aufgetreten ist.
- Datentyp:**
- ```
struct FreeList *freelist;  
ULONG anfang;  
int länge;  
int fehler;
```
- Sonstiges:** Dieser Befehl dient dazu, Speicherbereiche zu verwalten. Immer wenn ein Programm beendet ist, kann der zugehörige Speicherbereich mit AddFreeList in eine FreeList eingetragen werden. Diese wird vom System verwaltet, das durch solche FreeList's den freien Speicherplatz ermitteln kann.
- Wird ein neues Programm geladen, kann anhand einer FreeList ermittelt werden, an welcher Stelle im Speicher genügend freier Platz vorhanden ist.
- Die FreeList-Structure hat folgende Form:
- ```
struct FreeList {
 WORD fl_NumFree;
 struct List fl_MemList;
};
```
- fl\_NumFree gibt die Länge eines freien Speicherbereiches in 8Byte-Blocks an.
- fl\_MemList ist eine Unterstruktur, die mehr Einzelheiten über diesen Speicherbereich enthält.
- Referenz:** Siehe auch FreeFreeList

## 12.2 AllocWLObject

|            |                                                                                                                                                          |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | <code>objekt = AllocWLObject();</code>                                                                                                                   |
| Funktion:  | Erstellt eine WLObject-Structure, die für Info-Dateien benötigt wird.                                                                                    |
| Parameter: | Keine Parameter.                                                                                                                                         |
| Ergebnis:  | <code>objekt</code> -> Zeiger auf die WLObject-Structure.                                                                                                |
| Datentyp:  | <code>struct WLObject *objekt;</code>                                                                                                                    |
| Sonstiges: | Der Aufbau der WLObject-Structure ist dem Anhang zu entnehmen.<br><br>Dieser Befehl wird nur verwendet, wenn eine neue WLObject-Structure benötigt wird. |
| Referenz:  | Siehe auch <code>FreeWLObject</code>                                                                                                                     |

## 12.3 BumpRevision

- Syntax:**           länge = BumpRevision(neu,alt);
- Funktion:**       Erzeugt einen neuen Namen für eine kopierte Datei.
- Parameter:**    neu               -> Name der Dateikopie.  
                  alt               -> Name des Originals.
- Ergebnis:**     länge           -> Länge des Kopienamen.
- Datentyp:**     char neu[], alt[];  
                  int länge;
- Sonstiges:**     »neu« muß mindestens 31 Bytes lang sein, da ein Dateiname maximal 30 Zeichen lang sein darf und ein Nullbyte als Abschluß benötigt wird.
- Der Name der Kopie beginnt im Normalfall mit »copy of«. Nur wenn der Dateiname länger als 30 Zeichen wäre, wird dieser Text gekürzt.

## 12.4 FreeDiskObject

|            |                                                                                                                                                             |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | FreeDiskObject(diskobject);                                                                                                                                 |
| Funktion:  | Löscht den gesamten Speicher, der von einer Info-Datei belegt wird.                                                                                         |
| Parameter: | diskobject      -> Zeiger auf die DiskObject-Structure der Datei, dessen Info-Datei aus dem Speicher gelöscht werden soll.                                  |
| Ergebnis:  | Kein Ergebnis.                                                                                                                                              |
| Datentyp:  | struct DiskObject *diskobject;                                                                                                                              |
| Sonstiges: | Die Info-Datei enthält solche Daten wie Position des Icons und das Icon selbst.<br><br>Zur DiskObjekt-Structure gelangt man über den GetDisk-Object-Befehl. |
| Referenz:  | Für die DiskObject-Structure siehe Anhang B und GetDisk-Object                                                                                              |

## 12.5 FreeFreeList

- Syntax:** FreeFreeList(freelist);
- Funktion:** Löscht alle Einträge in der spezifizierten FreeList-Structure und die Structure selbst.
- Parameter:** freelist -> Zeiger auf die FreeList-Structure, die gelöscht werden soll.
- Ergebnis:** Kein Ergebnis.
- Datentyp:** struct FreeList \*freelist;
- Sonstiges:** Für weitere Informationen siehe AddFreeList.
- Referenz:** Siehe auch AddFreeList

## 12.6 FreeWbObject

|            |                                                                                                                                                                                                                                                                                                                                                                        |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | FreeWbObject(wbobject);                                                                                                                                                                                                                                                                                                                                                |
| Funktion:  | Dieser Befehl löscht alle Eintragungen in der spezifizierten WbObject-Structure und diese selbst.                                                                                                                                                                                                                                                                      |
| Parameter: | wbobject      -> Zeiger auf die WbObject-Structure eines Info-Files, die gelöscht werden soll.                                                                                                                                                                                                                                                                         |
| Ergebnis:  | Kein Ergebnis.                                                                                                                                                                                                                                                                                                                                                         |
| Datentyp:  | struct WbObject *wbobject;                                                                                                                                                                                                                                                                                                                                             |
| Sonstiges: | Dieser Befehl verwendet die AddFreeList-Funktion, um den freigewordenen Speicherplatz dem System bekanntzumachen.<br><br>FreeWbObject wird nur dann verwendet, wenn kein Bedarf mehr für eine, mit GetWbObject oder AllocWbObject eingeladenen, bzw. erstellten WbObject-Structure vorhanden ist. Diese enthält die Daten, die aus einer Info-Datei übernommen wurden. |
| Referenz:  | Siehe auch AllocWbObject, GetWbObject, PutWbObject                                                                                                                                                                                                                                                                                                                     |

## 12.7 GetDiskObject

- Syntax:** `diskobject = GetDiskObject(name);`
- Funktion:** Lädt die DiskObject-Structure einer Datei von Diskette.
- Parameter:** `name` -> Dateiname, deren DiskObject-Structure geladen werden soll.
- Ergebnis:** `diskobject` -> Zeiger auf die geladene DiskObject-Structure.
- Datentyp:** `char name[];`  
`struct DiskObject *diskobject;`
- Sonstiges:** Die DiskObject-Structure wird für einige Anwendungen, wie zum Beispiel für GetIcon, benötigt.
- Referenz:** Siehe auch PutDiskObject und GetIcon

## 12.8 GetIcon

- Syntax:**        `status = GetIcon(name,diskobject,freelist);`
- Funktion:**        Gleiche Funktion wie `GetDiskObject`.
- Parameter:**     `name`            -> Name der Datei, deren `DiskObject-Structure` geladen werden soll.
- `diskobject`        -> Zeiger auf eine leere `DiskObject-Structure`.
- `freelist`            -> Zeiger auf die `FreeList-Structure`, aus der der benötigte Speicherplatz entnommen werden soll.
- Ergebnis:**        `status`            -> ist 0, wenn ein Fehler aufgetreten ist.
- Datentyp:**        `char name[];`  
                  `struct DiskObject *diskobject;`  
                  `struct FreeList *freelist;`  
                  `int status;`
- Sonstiges:**        Diese Funktion hat die gleiche Aufgabe wie `GetDiskObject`. Allerdings muß hier der Programmierer mehr in das Geschehen eingreifen. Während sich `GetDiskObject` automatisch die benötigte `FreeList-Structure` »sucht«, muß sie hier angegeben werden.
- Referenz:**        Siehe auch `GetDiskObject` und `PutIcon`

## 12.9 GetWBOBJECT

- Syntax:**            `wbobject = GetWBOBJECT(name);`
- Funktion:**         Holt sämtliche Informationen, die in der WBOBJECT-Structure gespeichert werden, von Diskette.
- Parameter:**      `name`            -> Name der Datei, von der die Informationen geholt werden sollen.
- Ergebnis:**       `wbobject`        -> Zeiger auf die WBOBJECT-Structure, die nach dem Aufruf des Befehls die nötigen Informationen enthält.
- Datentyp:**        `char name[];`  
                      `struct WBOBJECT *wbobject;`
- Sonstiges:**        Dieser Befehl ist nur für Programmierer interessant, die direkt in die Workbench-Steuerung eingreifen wollen.
- Referenz:**         Siehe auch `PutWBOBJECT`

## 12.10 PutDiskObject

|            |                                                                                                                       |                                                                                                       |
|------------|-----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| Syntax:    | <code>status = PutDiskObject(name,diskobject);</code>                                                                 |                                                                                                       |
| Funktion:  | Speichert die Informationen der DiskObject-Structure in der Info-Datei einer Datei ab, die mit »name« bezeichnet ist. |                                                                                                       |
| Parameter: | <code>name</code>                                                                                                     | -> Name der Datei, in deren Info-Datei die Informationen gespeichert werden sollen.                   |
|            | <code>diskobject</code>                                                                                               | -> Zeiger auf die DiskObject-Structure, die die Informationen enthält, die gespeichert werden sollen. |
| Ergebnis:  | <code>status</code>                                                                                                   | -> ist 0, wenn ein Fehler aufgetreten ist.                                                            |
| Datentyp:  | <code>char name[];</code><br><code>struct DiskObject *diskobject;</code><br><code>int status;</code>                  |                                                                                                       |
| Sonstiges: | Gegenstück zu GetDiskObject.                                                                                          |                                                                                                       |
| Referenz:  | Siehe auch GetDiskObject                                                                                              |                                                                                                       |

## 12.11 PutIcon

|            |                                                                                                                       |                                                                                                       |
|------------|-----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| Syntax:    | <code>status = PutIcon(name,diskobject);</code>                                                                       |                                                                                                       |
| Funktion:  | Speichert die Informationen der DiskObject-Structure in der Info-Datei einer Datei ab, die mit »name« bezeichnet ist. |                                                                                                       |
| Parameter: | <code>name</code>                                                                                                     | -> Name der Datei, in deren Info-Datei die Informationen gespeichert werden sollen.                   |
|            | <code>diskobject</code>                                                                                               | -> Zeiger auf die DiskObject-Structure, die die Informationen enthält, die gespeichert werden sollen. |
| Ergebnis:  | <code>status</code>                                                                                                   | -> ist 0, wenn ein Fehler aufgetreten ist.                                                            |
| Datentyp:  | <code>char name[];</code><br><code>struct DiskObject *diskobject;</code><br><code>int status;</code>                  |                                                                                                       |
| Sonstiges: | Gegenstück zu GetIcon.                                                                                                |                                                                                                       |
| Referenz:  | Siehe auch GetIcon                                                                                                    |                                                                                                       |

## 12.12 PutWLObject

|            |                                                                                                                     |                                                                                                     |
|------------|---------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| Syntax:    | <code>status = PutWLObject(name,wbobject);</code>                                                                   |                                                                                                     |
| Funktion:  | Speichert die Informationen der WLObject-Structure in der Info-Datei einer Datei ab, die mit "name" bezeichnet ist. |                                                                                                     |
| Parameter: | name                                                                                                                | -> Name der Datei, in deren Info-Datei die Informationen gespeichert werden sollen.                 |
|            | wbobject                                                                                                            | -> Zeiger auf die WLObject-Structure, die die Informationen enthält, die gespeichert werden sollen. |
| Ergebnis:  | status                                                                                                              | -> ist 0, wenn ein Fehler aufgetreten ist.                                                          |
| Datentyp:  | <code>char name[];</code><br><code>struct WLObject *wbobject;</code><br><code>int status;</code>                    |                                                                                                     |
| Sonstiges: | Gegenstück zu GetWLObject.                                                                                          |                                                                                                     |
| Referenz:  | Siehe auch GetWLObject                                                                                              |                                                                                                     |



# Die Sprachausgabe

Die Sprachausgabe des Amiga wird über eine Translator-Library und eine Narrator-Device gesteuert.

Die Translator-Library enthält Befehle, um einen Text zu »übersetzen«, das heißt, in eine aussprechbare Form zu bringen.

Die Translator-Library wird folgendermaßen geöffnet:

```
struct Library *TranslatorBase;
main()
{
 TranslatorBase = OpenLibrary("translator.library",0);
 if(TranslatorBase == NULL) exit();
 .
 .
}
```

Anschließend kann der Library-Befehl Translate verwendet werden. Dieser hat folgende Form:

```
fehler = Translate(eingabe,einlänge,ausgabe,auslänge);
```

|          |    |                                                             |
|----------|----|-------------------------------------------------------------|
| eingabe  | -> | Zeiger auf den Text, der zu übersetzen ist.                 |
| einlänge | -> | Länge des zu übersetzenden Textes.                          |
| ausgabe  | -> | Puffer, in den der übersetzte Text eingetragen werden soll. |
| auslänge | -> | Länge des Puffers.                                          |
| fehler   | -> | ist > 0, wenn der Puffer nicht groß genug war.              |

Ist der Text mit Translate übersetzt worden, so muß er über Narrator-Device ausgegeben werden. Da sich die Narrator-Device auf Diskette befindet, muß sich die Diskette, die Narrator-Device im Devs-Verzeichnis enthält, in einem Laufwerk befinden.

Die Narrator-Device ist wie ein normales Device zu behandeln. Allerdings enthält sie eine spezielle Structure, die initialisiert werden muß. Diese Structure heißt »narrator\_rb«. Folgende Werte in ihr müssen deklariert werden:

```
message.io_Message.mn_ReplyPort - Antwort-Port
 message.io_Command - Kommando
 message.io_Data - Sprachtext
 message.io_Length - Länge des Sprachtextes
 rate - Worte pro Minute
 pitch - Grundfrequenz
 sex - Geschlecht
 ch_masks - Ausgabekanäle
 nm_masks - Anzahl der Kanäle
 volume - Lautstärke
```

Für »rate« dürfen Werte zwischen 40 und 400 eingegeben werden. »pitch« darf Werte zwischen 65 und 320 annehmen. Für das Geschlecht kann MALE oder FEMALE angegeben werden. Für die Kanäle kann 3, 5, 10, 12 oder jede Kombination davon angegeben werden.

Anschließend muß mit DoIO der Text ausgegeben werden.

```

1 /*****
2
3 Narrator / Demonstration
4 last update 26/05/87
5 von Frank Kremser und Joerg Koch
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Lets have fun. Der Amiga spricht zu Ihnen !!!!!
11 Experimentieren mit den Werten lohnt sich !!!!!
12
13 *****/
14
15 #include <exec/types.h> /* Includes die wir brauchen */
16 #include <exec/exec.h>
17 #include <intuition/intuition.h>
18 #include <devices/narrator.h>
19 #include <libraries/translator.h>
20
21 extern struct MsgPort *CreatePort(); /* Message-Port zuweisen */
22
23 ULONG TranslatorBase; /* TranslatorBase festlegen */
24
25 struct MsgPort *narratorPort;
26 struct narrator_rb request; /* Narrator-Structure */
27
28 UBYTE original[] = {"I AM YOUR AMIGA"}; /* Ausgebener Text !!! */
29 UBYTE translate[500]; /* Translator - Puffer */
30
31 BYTE Kanal[4] = {3,5,10,12}; /* Ausgabekanaele setzen */
32
33 main()
34 {
35 int fehler;
36
37 TranslatorBase = OpenLibrary("translator.library",0);
38 if(TranslatorBase == NULL) exit();
39
40 narratorPort = (struct MsgPort *)CreatePort("narrator.port",0);
41 if(narratorPort == NULL) exit();
42
43 fehler = OpenDevice("narrator.device",0,&request,0);
44 if(fehler != 0)
45 {
46 CloseLibrary(TranslatorBase); /* wenn Fehler */
47 DeletePort(narratorPort); /* Lib schliessen */
48 exit();
49 };
50
51 fehler = Translate(&original,sizeof(original)
52 ,translate,sizeof(translate));
53 if(fehler != 0) exit();
54
55 request.message.io_Message.mn_ReplyPort = narratorPort;
56 request.message.io_Command = CMD_WRITE; /* Kommando Schreiben */
57 request.message.io_Data = (APTR)translate; /* Pointer auf Text */
58 request.message.io_Length = sizeof(translate); /* Groesse des Textes */
59 request.rate = 120; /* Worte pro Minute: 40 < rate < 400 */
60 request.pitch = 230; /* 65 < pitch < 320 */
61 request.sex = FEMALE; /* Geschlecht : auch MALE */
62 request.ch_masks = Kanal; /* Kanaele */
63 request.nm_masks = sizeof(Kanal);
64 request.volume = 64; /* Lautstaerke max 64 */

```

```
65
66 fehler = DoIO(&request); /* Text aussprechen */
67
68 CloseDevice(&request); /* Device schliessen */
69 DeletePort(narratorPort); /* Port schliessen */
70 CloseLibrary(TranslatorBase); /* Lib schliessen */
71 }
```

# Multitasking

Rechner der neusten Generation, wie die Amiga-Serie, werden nicht nur durch ihr sehr gutes Preis-Leistungsverhältnis, sondern auch durch ihr hervorragendes Betriebssystem sehr hoch gewertet. Mit diesen neuen Betriebssystemen konnten auch erstmals Begriffe wie Multitasking in den Home- und Hobby-Computerbereich vordringen.

Unter Multitasking versteht man, daß ein Rechner mehrere Aufgaben zur »gleichen Zeit« verarbeiten bzw. ausführen kann (Multitasking -> multi = mehrere, tasks = Aufgaben). Dem Anwender erscheint es auf jeden Fall so, als würde der Rechner mehrere Programme gleichzeitig bearbeiten. In Wirklichkeit bearbeitet der Rechner beim Multitasking einen kleinen Teil eines Programmes, bricht es ab und beginnt mit der Bearbeitung eines weiteren Programmes, bricht dies wieder ab und führt die Bearbeitung des ersten Programmes an der Abbruchstelle fort. Der Rechner springt also zwischen den Programmen hin und her. Dies geschieht so schnell, daß der Anwender meint, die Programme würden parallel ablaufen. Diese Methode wird auch als »Scheduling« bezeichnet.

Voraussetzungen zum Multitasking sind einmal ein schneller Prozessor, der Amiga besitzt den MC68000 von Motorola, viel Speicher, da die Tasks verwaltet werden müssen, und ein Betriebssystemkern, der multitaskingfähig ist. Darunter versteht man, daß dieses Betriebssystem die Verwaltung der Arbeitsprozesse (Tasks) übernehmen kann.

Das Amiga-Betriebssystem besitzt einen hervorragenden Kern, dieser wird EXEC (executive - ausführend) genannt. EXEC übernimmt die komplette »Kontrolle« des MC68000, teilt die Zeit für verschiedene Tasks ein, übernimmt Ein- und Ausgabefunktionen, die Vermittlung von Nachrichten und noch einiges mehr. EXEC ist, aufgrund seiner Wichtigkeit, das einzige Library, das beim Amiga jederzeit geöffnet ist. Nun aber zurück zum Multitasking.

Zwei Möglichkeiten die Multitaskingfähigkeiten des Amiga zu entlocken, hat bestimmt schon jeder Amiga-Besitzer kennengelernt. Zunächst ist da das Starten von Programmen von der Workbench aus. Jedes Programm, das durch Anklicken der Maustaste aufgerufen wird, wird als separater Task behandelt. Eine weitere Möglichkeit bietet das CLI: durch den Befehl »RUN« kann ein Programm als separater Task gestartet werden. Die dritte Möglichkeit ist das Starten von Tasks in eigenen Programmen. So kann man z.B. zwei Windows öffnen und »gleichzeitig« in beiden verschiedene Linien Zeichen oder Berechnungen als Tasks ausführen. Dem Multitasking ist hier nur in der Anzahl und in dem vorhandenen Speicherplatz eine Grenze gesetzt, denn je mehr Tasks, desto mehr Speicherplatz und desto langsamer wird das System.

Multitasking in eigenen Programmen setzt nach der Meinung von Experten gewisse Grundkenntnisse des Betriebssystemkerns EXEC und des Multitaskingsystems voraus. Das EXEC Support Library enthält jedoch Funktionen, die den Umgang mit Multitasking in eigenen Programmen erheblich vereinfachen. Der Anwender braucht somit kaum Vorkenntnisse, jedoch werden die Möglichkeiten des Multitasking nicht vollkommen ausgeschöpft.

Bei Verwendung des Multitasking in eigenen Programmen werden neben der Hauptroutine verschiedene Unterroutinen benötigt, die mit CreateTask als separate Tasks aufgerufen werden können. Die Funktion CreateTask übernimmt für uns alle wichtigen Einrichtarbeiten, die zum »Anheften« von Tasks an das System notwendig sind. Mit CreateTask muß auch eine Priorität des jeweiligen Task übergeben werden, damit ist die Wichtigkeit des jeweiligen Tasks gemeint. Sie muß mit der Priorität der Hauptroutine abgestimmt sein. Ist z.B. die Priorität des Tasks kleiner als die der Hauptroutine, so wird der Task erst gar nicht ausgeführt. Die Task-Priorität kann einen Wert von -128 bis +127 annehmen, wobei -128 die niedrigste Priorität darstellt. Die meisten Tasks des Amiga-Systems liegen im Bereich von -20 bis +20. Soll die Priorität eines Programms nachträglich geändert werden, so kann das mit der Funktion ChangePri() geschehen. Nachdem der Task seine Aufgabe erfüllt hat, muß er wieder aus der Task-Liste des Systems mit DeleteTask() oder RemTask() gelöscht werden.

## 14.1 ChangePri

|            |                                                                                                                                        |                                                                                                                         |
|------------|----------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | ChangePri(Task, Priorität);                                                                                                            |                                                                                                                         |
| Funktion:  | Ändert die Priorität eines Tasks.                                                                                                      |                                                                                                                         |
| Parameter: | Task                                                                                                                                   | -> Zeiger auf die Task-Structure des Tasks, dessen Priorität geändert werden soll.                                      |
|            | Priorität                                                                                                                              | -> Die neue Priorität des Tasks. Der Wert kann zwischen -128, niedrigste Priorität, und +128, höchste Priorität liegen. |
| Ergebnis:  | Kein Ergebnis.                                                                                                                         |                                                                                                                         |
| Datentyp:  | struct Task *Task;<br>BYTE Priorität;                                                                                                  |                                                                                                                         |
| Sonstiges: | Die Priorität der Tasks muß, wenn sie von einem Hauptprogramm aufgerufen wird, mit der Priorität des Hauptprogramms abgestimmt werden. |                                                                                                                         |

## 14.2 CreateTask

|            |                                                                                                                                                                                                                                                         |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | <code>Task = CreateTask(&amp;Name[0], Priorität, &amp;Funktion[0], Stack);</code>                                                                                                                                                                       |
| Funktion:  | Bildet einen Task.                                                                                                                                                                                                                                      |
| Parameter: | <code>&amp;Name[0]</code> -> Beliebiger Name für den Task.<br><code>Priorität</code> -> Priorität des Tasks.<br><code>&amp;Funktion[0]</code> -> Die Unterroutine, die als Task aufgerufen werden soll.<br><code>Stack</code> -> Speicher für den Task. |
| Ergebnis:  | <code>Task</code> -> Zeiger auf die Task-Structure des neuen Tasks. Ist sie NULL, so konnte kein neuer Task erzeugt werden.                                                                                                                             |
| Datentyp:  | <code>char Name[];</code><br><code>BYTE Priorität;</code><br><code>char Funktion[];</code><br><code>LONG Stack;</code><br><code>struct Task *Task;</code>                                                                                               |
| Sonstiges: | Die Priorität des Tasks kann später mit <code>ChangePri</code> geändert werden. Der Speicher für den Task ist von der Größe des jeweiligen Programms abhängig. Ein Speicher von 20000 Bytes reicht aber in jedem Fall aus.                              |
| Referenz:  | Eine Anwendungsmöglichkeit finden Sie in dem Programm <code>Multidemo</code> .                                                                                                                                                                          |

## 14.3 DeleteTask

|            |                                                                                         |
|------------|-----------------------------------------------------------------------------------------|
| Syntax:    | DeleteTask(Task);                                                                       |
| Funktion:  | Löscht einen Task.                                                                      |
| Parameter: | Task           -> Zeiger auf die Task-Structure des<br>Tasks, der gelöscht werden soll. |
| Ergebnis:  | Kein Ergebnis.                                                                          |
| Datentyp:  | struct Task *Task;                                                                      |
| Sonstiges: | Ein Task kann auch mit der Funktion RemTask() gelöscht<br>werden.                       |
| Referenz:  | Siehe auch RemTask                                                                      |

## 14.4 RemTask

|                   |                                                                                              |
|-------------------|----------------------------------------------------------------------------------------------|
| <b>Syntax:</b>    | RemTask(Task);                                                                               |
| <b>Funktion:</b>  | Löscht einen Task.                                                                           |
| <b>Parameter:</b> | Task                   -> Zeiger auf die Task-Structure des Tasks, der gelöscht werden soll. |
| <b>Ergebnis:</b>  | Kein Ergebnis.                                                                               |
| <b>Datentyp:</b>  | struct Task *Task;                                                                           |
| <b>Sonstiges:</b> | Ein Task kann auch mit DeleteTask gelöscht werden.                                           |
| <b>Referenz:</b>  | Siehe auch DeleteTask                                                                        |

```

1 /*****
2
3 Multitasking-Demonstration
4 last update 26/05/87
5 von Joerg Koch und Frank Kremser
6 (c) Markt und Technik 1987
7
8 *****/
9
10 Einfache Multitasking-Demo. Zwei Unterfunktionen werden als Tasks
11 gestartet.
12
13 *****/
14
15 #include <exec/types.h> /* Include-Files laden */
16 #include <exec/tasks.h>
17 #include <exec/libraries.h>
18 #include <exec/devices.h>
19 #include <exec/memory.h>
20 #include <exec/execbase.h>
21 #include <exec/exec.h>
22 #include <devices/keymap.h>
23 #include <graphics/copper.h>
24 #include <graphics/display.h>
25 #include <graphics/gfxbase.h>
26 #include <graphics/text.h>
27 #include <graphics/view.h>
28 #include <graphics/gels.h>
29 #include <graphics/regions.h>
30 #include <graphics/sprite.h>
31 #include <hardware/blit.h>
32 #include <intuition/intuition.h>
33 #include <intuition/intuitionbase.h>
34
35
36 struct IntuitionBase *IntuitionBase; /* Lib - Zeiger */
37 struct GfxBase *GfxBase;
38 struct ExecBase *ExecBase;
39
40 int ende1, ende2;
41
42 struct Task *t1, *t2; /* Task - Zeiger */

```

```
43
44 struct Window *w1, *w2; /* Window - Zeiger */
45
46 struct NewWindow nw1 = /* Window fuer Task 1 definieren */
47 {
48 30,
49 30,
50 280,
51 200,
52 2,
53 1,
54 NULL,
55 NULL,
56 NULL,
57 NULL,
58 "Task Nummer 1",
59 NULL,
60 NULL,
61 0,
62 0,
63 0,
64 0,
65 WBENCHSCREEN
66 };
67
68 struct NewWindow nw2 = /* Window fuer Task2 definieren */
69 {
70 350,
71 30,
72 280,
73 200,
74 2,
75 1,
76 NULL,
77 NULL,
78 NULL,
79 NULL,
80 "Task Nummer 2",
81 NULL,
82 NULL,
83 0,
84 0,
85 0,
86 0,
87 WBENCHSCREEN
88 };
89
90
91 task1() /* Unterfunktion */
92 { /* wird als Task 1 gestartet */
93 LONG warte;
94 int schleife;
95
96 w1 = (struct Window *)OpenWindow(&nw1); /* Window oeffnen */
97 SetDrMd(w1->RPort,JAM1);
98 SetAPen(w1->RPort,3);
99 for(schleife = 10; schleife<200; schleife=schleife+2)
100 {
101 Move(w1->RPort,0,schleife); /* Gitter zeichnen */
102 Draw(w1->RPort,280,schleife);
103 };
104
105 for(schleife = 0; schleife<280; schleife=schleife+2)
```

```
106 {
107 Move(w1->RPort,schleife,10); /* Gitter zeichnen */
108 Draw(w1->RPort,schleife,200);
109 };
110
111 for(warte = 0; warte < 50000; warte++);
112 CloseWindow(w1); /* Task fertig, Window */
113 ende1 = 1; /* schliessen */
114 }
115
116 task2()
117 {
118 LONG warte;
119 int schleife;
120
121 w2 = (struct Window *)OpenWindow(&nw2); /* Window Task2 oeffnen */
122 SetDrMd(w2->RPort,JAM1);
123 SetAPen(w2->RPort,1);
124 for(schleife = 10; schleife<200; schleife=schleife+2)
125 {
126 Move(w2->RPort,0,schleife); /* Gitter zeichnen */
127 Draw(w2->RPort,280,schleife);
128 };
129
130 for(schleife = 0; schleife<280; schleife=schleife+2)
131 {
132 Move(w2->RPort,schleife,10);
133 Draw(w2->RPort,schleife,200); /* Gitter zeichnen */
134 };
135
136 for(warte = 0; warte < 50000; warte++); /* Task fertig, */
137 CloseWindow(w2); /* Window schliessen */
138 ende2 = 2;
139 }
140
141
142 main() /* Hauptprogramm */
143 {
144 LONG warte;
145
146 ende1 = ende2 = 0; /* ExecBase oeffnen */
147
148 ExecBase = (struct ExecBase *)OpenLibrary("exec.library",0);
149
150 IntuitionBase = (struct IntuitionBase *) /* Intui-Lib oeffnen */
151 OpenLibrary("intuition.library",0);
152
153 GfxBase = (struct GfxBase *)
154 OpenLibrary("graphics.library",0);
155 /* Task 1 und Task 2 bilden und */
156 /* starten */
157
158 t2 = (struct Task *)CreateTask("Task 2",0,task2,5000);
159 t1 = (struct Task *)CreateTask("Task 1",0,task1,5000);
160
161 while((ende1 == 0) && (ende2 == 0)); /* beide fertig, dann Ende */
162
163 for(warte = 0; warte < 50000; warte++);
164
165
166 CloseLibrary(IntuitionBase); /* Libs schliessen */
167 CloseLibrary(GfxBase);
168 CloseLibrary(ExecBase);
169 }
```

## Mathematik-Libraries

Der Amiga stellt verschiedene Bibliotheken mit mathematischen Befehlen bereit. Zwar sind viele dieser Befehle auch schon von »C« aus vorhanden, doch benötigen diese ein Vielfaches der Zeit. Dies trifft vor allen Dingen auf die Funktionen des Lattice-C-Compilers zu, da dieser sämtliche Gleitkommazahlen mit doppelter Genauigkeit berechnet.

Dieser Punkt bringt noch weitere Schwierigkeiten mit sich. Da der Lattice-C-Compiler sämtliche Gleitkomma-Zahlen mit doppelter Genauigkeit speichert und berechnet, treten Probleme auf, wenn die mathematischen Befehle, die mit einfacher Genauigkeit rechnen, verwendet werden. In diesem Fall werden nämlich sämtliche Ergebnisse falsch interpretiert. Um dieses Problem zu umgehen, muß ein Trick angewendet werden. Dies gilt, wie schon gesagt, nur für den Lattice-C-Compiler. Die Besitzer eines Aztec-Compilers können für sämtliche Funktionen, die FFP-Zahlen verarbeiten, FLOAT-Werte angeben. Das bedeutet, sämtliche »ffp«-Variablen können mit »FLOAT ffp1, ffp2, ...usw.« deklariert werden.

Die Besitzer eines Lattice-C-Compiler müssen folgenden Trick anwenden:

```
main()
{
 union test
 {
 FLOAT ffp;
 int int;
 }
 /* Deklarieren einer Structure, bestehend */
 /* aus einer FLOAT- und einer Integer- */
 /* Variablen. »union« bewirkt, das beide */
 /* Elemente der Structure an der relativen */
 /* zahl */
 /* Adresse 0 beginnen. Der Name der Structure */
 /* ist »zahl« */

 zahl.ffp = 0.7; /* Variable belegen */
 zahl.int = SPFieee(zahl.int)
 /* Variable in FFP-Format wandeln */
 xyz = SPSqrt(zahl.int); /* Nun kann gerechnet werden */
}
```

Diese Methode ist zwar recht umständlich, ist aber leider nicht zu vermeiden. Dafür steigt die Rechengeschwindigkeit enorm an.

Besitzer neuerer Lattice-Versionen können dieses Problem beseitigen, indem Sie einen Compiler-Schalter setzen. Ziehen Sie dazu bitte Ihr Handbuch zu Rate.

## 15.1 Mathematische Grundfunktionen

Die mathematischen Grundfunktionen befinden sich in der »mathffp«. Um sie verwendet zu können, muß die Include-Datei »math.h« eingelesen werden. Im Anschluß daran müssen die Funktionen extern bereitgestellt werden. Dazu dient die »extern«-Funktion. Dies müssen ausschließlich die Lattice-Besitzer einfügen, da dies mit den oben genannten Problemen in Verbindung steht. Anschließend muß die »mathffp«-Library geöffnet werden. Ist dies geschehen, so können die Funktionen verwendet werden. Im Programm könnte das folgendermaßen aussehen:

```
#include <exec/types.h>
#include <math.h>

extern int SPFix; /* Nur für Lattice-Besitzer */
extern int SPAbs;
usw.
extern int SPDIV;

main()
{
 union test /* Für Lattice-Besitzer */
 {
 FLOAT ffp;
 int int;
 } zahl;

 FLOAT ffp; /* Für Aztec-Besitzer */

 ULONG MathBase;

 /* Library öffnen */
 if((MathBase = OpenLibrary("mathffp.library",0)) == 0) exit();
 /* Berechnungen s.o. */
 CloseLibrary(MathBase);
}
```

Nach diesem Schema können Sie also vorgehen. Wollen Sie Ihre Berechnungen mit doppelter Genauigkeit ermitteln, so können Sie die »mathieeedoubbas.library« verwenden. Diese Befehle können sehr einfach verwendet werden. Dazu gehen wir folgendermaßen vor:

```
#include <exec/types.h>
#include <math.h>

main()
{
 double ffp1, ffp2;

 ULONG MathIeeeDoubBasBase;

 /* Library öffnen */
```

```

if((MathBase = OpenLibrary("mathieeedoubbas.library",0)) == 0)
 exit();

ffp1 = -23.4567; /* Variable belegen */
ffp2 = IEEEEDPNeg(ffp1); /* Berechnung */

CloseLibrary(MathIeeeDoubBasBase);
}

```

Diese Befehle mit doppelter Genauigkeit haben wir nicht extra aufgeführt, da sie mit den nachfolgenden Befehlen mit einfacher Genauigkeit fast übereinstimmen. Wenn der Befehl mit einfacher Genauigkeit die Form »SPSub« hat, so hat der Befehl mit doppelter Genauigkeit die Form »IEEEEDPSub«.

### 15.1.1 SPFix

**Syntax:** int = SPFix(ffp);

**Funktion:** Wandelt eine FFP- in eine Integer-Zahl um.

**Parameter:** ffp -> FFP-Zahl, die umgewandelt werden soll.

**Ergebnis:** int -> Integer-Zahl.

**Datentyp:** FLOAT ffp;  
int int;

**Sonstiges:** Diese Funktion hat die gleiche Wirkung wie die Absolutfunktion.

Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15.

**Referenz:** Siehe auch SPAbs

### 15.1.2 SPFlt

**Syntax:** ffp = SPFlt(int);

**Funktion:** Wandelt eine Integer- in eine FFP-Zahl um.

**Parameter:** int -> Integer-Zahl, die umgewandelt werden soll.

**Ergebnis:** ffp -> FFP-Zahl.

**Datentyp:** int int;  
FLOAT ffp;

Sonstiges: Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15.

### 15.1.3 SPCmp

Syntax: `int = SPCmp(ffp1, ffp2);`

Funktion: Vergleicht zwei FFP-Zahlen miteinander.

Parameter: `ffp1, ffp2` -> FFP-Zahlen, die verglichen werden sollen.

Ergebnis: `int` -> Wenn `ffp1` kleiner als `ffp2` ist, ist `int` gleich `-1`, ist `ffp1` größer als `ffp2`, so ist `int` gleich `+1`. Sollten beide FFP-Zahlen gleich sein, so ist `int` gleich `0`.

Datentyp: `FLOAT ffp1, ffp2;`  
`int int;`

Sonstiges: Die Funktion SPCmp liefert das gleiche Ergebnis wie eine Signum-Funktion.

Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15.

### 15.1.4 SPTst

Syntax: `int = SPTst(ffp);`

Funktion: Testet eine FFP-Variable auf 0.

Parameter: `ffp` -> Zahl, die auf 0 getestet werden soll.

Ergebnis: `int` -> Wenn `ffp1` kleiner als `0` ist, ist `int` gleich `-1`, ist `ffp1` größer als `0`, so ist `int` gleich `+1`. Sollte die FFP-Zahl gleich `0` sein, so ist `int` gleich `0`.

Datentyp: `FLOAT ffp;`  
`int int;`

Sonstiges: Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15.

### 15.1.5 SPAbs

|            |                                                                                                                                          |                                                       |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------|
| Syntax:    | ffp1 = SPAbs(ffp2);                                                                                                                      |                                                       |
| Funktion:  | Ermittelt den Absolutwert einer FFP-Zahl.                                                                                                |                                                       |
| Parameter: | ffp2                                                                                                                                     | -> FFP-Zahl, deren Absolutwert ermittelt werden soll. |
| Ergebnis:  | ffp1                                                                                                                                     | -> Absolutwert von ffp2.                              |
| Datentyp:  | FLOAT ffp2;<br>FLOAT ffp1;                                                                                                               |                                                       |
| Sonstiges: | Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |                                                       |
| Referenz:  | Siehe auch SPFix                                                                                                                         |                                                       |

### 15.1.6 SPNeg

|            |                                                                                                                                                                                                                                                                          |                                                 |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| Syntax:    | ffp1 = SPNeg(ffp2);                                                                                                                                                                                                                                                      |                                                 |
| Funktion:  | Ermittelt das Negat einer FFP-Zahl.                                                                                                                                                                                                                                      |                                                 |
| Parameter: | ffp2                                                                                                                                                                                                                                                                     | -> FFP-Zahl, deren Negat ermittelt werden soll. |
| Ergebnis:  | ffp1                                                                                                                                                                                                                                                                     | -> Negat der FFP-Zahl.                          |
| Datentyp:  | FLOAT ffp2;<br>FLOAT ffp1;                                                                                                                                                                                                                                               |                                                 |
| Sonstiges: | Das Negat bedeutet, daß eine positive Zahl negativ wird und umgekehrt. Auf die Zahl 0 hat diese Funktion keinen Einfluß.<br><br>Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |                                                 |

### 15.1.7 SPAdd

|            |                           |                                           |
|------------|---------------------------|-------------------------------------------|
| Syntax:    | ffp1 = SPAdd(ffp2, ffp3); |                                           |
| Funktion:  | Addiert zwei FFP-Zahlen.  |                                           |
| Parameter: | ffp2, ffp3                | -> FFP-Zahlen, die addiert werden sollen. |
| Ergebnis:  | ffp1                      | -> Summe der beiden FFP-Zahlen.           |

|            |                                                                                                                                          |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|
| Datentyp:  | FLOAT ffp2, ffp3;<br>FLOAT ffp1;                                                                                                         |
| Sonstiges: | Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |
| Referenz:  | Siehe auch SPSub                                                                                                                         |

### 15.1.8 SPSub

|            |                                                                                                                                          |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | $ffp1 = SPSub(ffp2, ffp3);$                                                                                                              |
| Funktion:  | Subtrahiert zwei FFP-Variablen voneinander.                                                                                              |
| Parameter: | ffp2           -> FFP-Zahl, von der subtrahiert werden soll.<br>ffp3           -> FFP-Zahl, die subtrahiert werden soll.                 |
| Ergebnis:  | ffp1           -> Differenz zwischen den beiden FFP-Zahlen.                                                                              |
| Datentyp:  | FLOAT ffp2, ffp3;<br>FLOAT ffp1;                                                                                                         |
| Sonstiges: | Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |
| Referenz:  | Siehe auch SPAdd                                                                                                                         |

### 15.1.9 SPMul

|            |                                                                                                                                          |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | $ffp1 = SPMul(ffp2, ffp3);$                                                                                                              |
| Funktion:  | Multipliziert zwei FFP-Zahlen miteinander.                                                                                               |
| Parameter: | ffp2, ffp3       -> FFP-Zahlen, die miteinander multipliziert werden sollen.                                                             |
| Ergebnis:  | ffp1           -> Produkt der beiden FFP-Zahlen.                                                                                         |
| Datentyp:  | FLOAT ffp2, ffp3;<br>FLOAT ffp1;                                                                                                         |
| Sonstiges: | Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |
| Referenz:  | Siehe auch SPDiv                                                                                                                         |

### 15.1.10 SPDiv

|            |                                                                                                                                          |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | <code>ffp1 = SPDiv(ffp2, ffp3);</code>                                                                                                   |
| Funktion:  | Dividiert zwei FFP-Variablen.                                                                                                            |
| Parameter: | <code>ffp2</code> -> FFP-Zahl, die dividiert werden soll.<br><code>ffp3</code> -> FFP-Zahl, durch die dividiert werden soll.             |
| Ergebnis:  | <code>ffp1</code> -> Ergebnis der Division.                                                                                              |
| Datentyp:  | <code>FLOAT ffp2, ffp3;</code><br><code>FLOAT ffp1;</code>                                                                               |
| Sonstiges: | Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |
| Referenz:  | Siehe auch SPMul                                                                                                                         |

## 15.2 Transzendente Funktionen

Auf den nachfolgenden Seiten werden die Funktionen aufgeführt, die über die Grundfunktionen hinausgehen. Für die FFP-Werte, mit denen sie rechnen, gelten die gleichen Bedingungen, wie für die Grundfunktionen. Lesen Sie also bitte auch die Kapitel 15 und 15.1. Um die Befehle verwenden zu können muß zusätzlich zur »mathffp«-Library noch die »mathtrans«-Library geöffnet werden. Im Programm könnte das folgendermaßen aussehen:

```
#include <exec/types.h>
#include <math.h>

extern int SPAsin; /* Nur für Lattice-Besitzer */
extern int SPAcos;
.
.
usw.
.
.
extern int SPFieee;

main()
{
 union test /* Für Lattice-Besitzer */
 {
 FLOAT ffp;
 int int;
 } zahl;

 FLOAT ffp; /* Für Aztec-Besitzer */

 ULONG MathBase;
 ULONG MathTransBase;

 /* Libraries öffnen */
 if((MathBase = OpenLibrary("mathffp.library",0)) == 0) exit();

 if((MathTransBase = OpenLibrary("mathtrans.library",0)) == 0)
 exit();
 .
 .
 /* Berechnungen s.o. */
 .
 .
 CloseLibrary(MathTransBase);
 CloseLibrary(MathBase);
}
```

### 15.2.1 SPAsin

|            |                                                                                                                                          |                                                                |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------|
| Syntax:    | ffp1 = SPAsin(ffp2);                                                                                                                     |                                                                |
| Funktion:  | Ermittelt den Arcussinus einer FFP-Zahl.                                                                                                 |                                                                |
| Parameter: | ffp2                                                                                                                                     | -> Sinuswert, dessen zugehöriger Winkel ermittelt werden soll. |
| Ergebnis:  | ffp1                                                                                                                                     | -> Winkel im Bogenmaß (Radiant).                               |
| Datentyp:  | FLOAT ffp2;<br>FLOAT ffp1;                                                                                                               |                                                                |
| Sonstiges: | Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |                                                                |
| Referenz:  | Siehe auch SPAcos und SPAtan                                                                                                             |                                                                |

### 15.2.2 SPAcos

|            |                                                                                                                                          |                                                                  |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|
| Syntax:    | ffp1 = SPAcos(ffp2);                                                                                                                     |                                                                  |
| Funktion:  | Ermittelt den Arcuscosinus einer FFP-Zahl.                                                                                               |                                                                  |
| Parameter: | ffp2                                                                                                                                     | -> Cosinuswert, dessen zugehöriger Winkel ermittelt werden soll. |
| Ergebnis:  | ffp1                                                                                                                                     | -> Winkel im Bogenmaß (Radiant).                                 |
| Datentyp:  | FLOAT ffp2;<br>FLOAT ffp1;                                                                                                               |                                                                  |
| Sonstiges: | Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |                                                                  |
| Referenz:  | Siehe auch SPAsin und SPAtan                                                                                                             |                                                                  |

### 15.2.3 SPAtan

|            |                                            |                                                                  |
|------------|--------------------------------------------|------------------------------------------------------------------|
| Syntax:    | ffp1 = SPAtan(ffp2);                       |                                                                  |
| Funktion:  | Ermittelt den Arcustangens einer FFP-Zahl. |                                                                  |
| Parameter: | ffp2                                       | -> Tangenswert, dessen zugehöriger Winkel ermittelt werden soll. |
| Ergebnis:  | ffp1                                       | -> Winkel im Bogenmaß (Radiant).                                 |

- Datentyp:** FLOAT ffp2;  
FLOAT ffp1;
- Sonstiges:** Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15.
- Referenz:** Siehe auch SPASin und SPACos

### 15.2.4 SPSin

- Syntax:** ffp1 = SPSin(ffp2);
- Funktion:** Ermittelt den Sinus einer FFP-Zahl.
- Parameter:** ffp2 -> Winkel im Bogenmaß, dessen Sinus ermittelt werden soll.
- Ergebnis:** ffp1 -> Sinuswert des Winkels.
- Datentyp:** FLOAT ffp2;  
FLOAT ffp1;
- Sonstiges:** Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15.
- Referenz:** Siehe auch SPCos und SPTan

### 15.2.5 SPCos

- Syntax:** ffp1 = SPCos(ffp2);
- Funktion:** Ermittelt den Cosinus einer FFP-Zahl.
- Parameter:** ffp2 -> Winkel im Bogenmaß, dessen Cosinus ermittelt werden soll.
- Ergebnis:** ffp1 -> Cosinuswert des Winkels.
- Datentyp:** FLOAT ffp2;  
FLOAT ffp1;
- Sonstiges:** Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15.
- Referenz:** Siehe auch SPSin und SPTan

## 15.2.6 SPTan

|            |                                                                                                                                          |                                                              |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|
| Syntax:    | ffp1 = SPTan(ffp2);                                                                                                                      |                                                              |
| Funktion:  | Ermittelt den Tangens einer FFP-Zahl.                                                                                                    |                                                              |
| Parameter: | ffp2                                                                                                                                     | -> Winkel im Bogenmaß, dessen Tangens ermittelt werden soll. |
| Ergebnis:  | ffp1                                                                                                                                     | -> Tangenswert des Winkels.                                  |
| Datentyp:  | FLOAT ffp2;<br>FLOAT ffp1;                                                                                                               |                                                              |
| Sonstiges: | Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |                                                              |
| Referenz:  | Siehe auch SPSin und SPCos                                                                                                               |                                                              |

## 15.2.7 SPSincos

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                                                  |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| Syntax:    | ffp1 = SPSincos(&ffp2, ffp3);                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                                  |
| Funktion:  | Ermittelt Sinus und Cosinus einer FFP-Zahl.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                  |
| Parameter: | ffp3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | -> Winkel im Bogenmaß, dessen Sinus und Cosinus ermittelt werden soll.           |
|            | &ffp2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | -> Zeiger auf eine FFP-Variable, in der der Cosinuswert gespeichert werden soll. |
| Ergebnis:  | ffp1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | -> Sinuswert des Winkels.                                                        |
| Datentyp:  | FLOAT ffp2, ffp3;<br>FLOAT ffp1;                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                                                  |
| Sonstiges: | <p>Da eine Funktion höchstens einen Wert direkt zurückgeben kann –hier ffp1–, muß der zweite Wert in einer Variablen gespeichert werden. Um dies zu können, muß der Funktion »gezeigt« werden, an welcher Stelle sie dies tun soll. Dazu dient »&amp;ffp2«. »&amp;« ermittelt die Adresse der Variablen »ffp2« und übergibt diese der Funktion, die an dieser Stelle den Cosinuswert speichert.</p> <p>Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15.</p> |                                                                                  |
| Referenz:  | Siehe auch SPSin und SPCos                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                  |

## 15.2.8 SPSinh

|            |                                                                                                                                          |                                                                |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------|
| Syntax:    | ffp1 = SPSinh(ffp2);                                                                                                                     |                                                                |
| Funktion:  | Ermittelt den hyperbolischen Sinus einer FFP-Zahl.                                                                                       |                                                                |
| Parameter: | ffp2                                                                                                                                     | -> FFP-Zahl, deren hyperbolischer Sinus ermittelt werden soll. |
| Ergebnis:  | ffp1                                                                                                                                     | -> enthält den hyperbolischen Sinus.                           |
| Datentyp:  | FLOAT ffp2;<br>FLOAT ffp1;                                                                                                               |                                                                |
| Sonstiges: | Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |                                                                |
| Referenz:  | Siehe auch SPCosh und SPTanh                                                                                                             |                                                                |

## 15.2.9 SPCosh

|            |                                                                                                                                          |                                                                  |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|
| Syntax:    | ffp1 = SPCosh(ffp2);                                                                                                                     |                                                                  |
| Funktion:  | Ermittelt den hyperbolischen Cosinus einer FFP-Zahl.                                                                                     |                                                                  |
| Parameter: | ffp2                                                                                                                                     | -> FFP-Zahl, deren hyperbolischer Cosinus ermittelt werden soll. |
| Ergebnis:  | ffp1                                                                                                                                     | -> enthält den hyperbolischen Cosinus.                           |
| Datentyp:  | FLOAT ffp2;<br>FLOAT ffp1;                                                                                                               |                                                                  |
| Sonstiges: | Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |                                                                  |
| Referenz:  | Siehe auch SPSinh und SPTanh                                                                                                             |                                                                  |

## 15.2.10 SPTanh

|            |                                                  |                                                                  |
|------------|--------------------------------------------------|------------------------------------------------------------------|
| Syntax:    | ffp1 = SPTanh(ffp2);                             |                                                                  |
| Funktion:  | Ermittelt den hyperbolischen Tangens einer Zahl. |                                                                  |
| Parameter: | ffp2                                             | -> FFP-Zahl, deren hyperbolischer Tangens ermittelt werden soll. |
| Ergebnis:  | ffp1                                             | -> enthält den hyperbolischen Tangens.                           |

- Datentyp:     FLOAT ffp2;  
                   FLOAT ffp1;
- Sonstiges:     Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15.
- Referenz:      Siehe auch SPSinh und SPCosh

### 15.2.11 SPExp

- Syntax:        ffp1 = SPExp(ffp2);
- Funktion:       Ermittelt den Wert von e hoch einer FFP-Zahl.
- Parameter:     ffp2               -> FFP-Zahl, mit der e potenziert werden soll.
- Ergebnis:     ffp1               -> Ergebnis von e hoch ffp2.
- Datentyp:     FLOAT ffp2;  
                   FLOAT ffp1;
- Sonstiges:     e ist gleich 2,718281828459045.  
                   Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15.
- Referenz:      Siehe auch SPLog

### 15.2.12 SPLog

- Syntax:        ffp1 = SPLog(ffp2);
- Funktion:       Ermittelt den natürlichen Logarithmus einer FFP-Zahl.
- Parameter:     ffp2               -> FFP-Zahl, deren natürlicher Logarithmus ermittelt werden soll.
- Ergebnis:     ffp1               -> enthält den natürlichen Logarithmus von ffp2.
- Datentyp:     FLOAT ffp2;  
                   FLOAT ffp1;
- Sonstiges:     Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15.
- Referenz:      Siehe auch SPExp und SPLog10

### 15.2.13 SPLog10

|            |                                                                                                                                          |                                                                    |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|
| Syntax:    | ffp1 = SPLog10(ffp2);                                                                                                                    |                                                                    |
| Funktion:  | Ermittelt den 10er-Logarithmus einer FFP-Zahl.                                                                                           |                                                                    |
| Parameter: | ffp2                                                                                                                                     | -> FFP-Zahl, deren Logarithmus zur Basis 10 ermittelt werden soll. |
| Ergebnis:  | ffp1                                                                                                                                     | -> enthält den Logarithmus zur Basis 10 von ffp2.                  |
| Datentyp:  | FLOAT ffp2;<br>FLOAT ffp1;                                                                                                               |                                                                    |
| Sonstiges: | Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |                                                                    |
| Referenz:  | Siehe auch SPLog                                                                                                                         |                                                                    |

### 15.2.14 SPPow

|            |                                                                                                                                          |                                                  |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|
| Syntax:    | ffp1 = SPPow(ffp2, ffp3);                                                                                                                |                                                  |
| Funktion:  | Ermittelt den Wert einer FFP-Zahl hoch einer weiteren FFP-Zahl.                                                                          |                                                  |
| Parameter: | ffp2                                                                                                                                     | -> Basis der Potenzierung.                       |
|            | ffp3                                                                                                                                     | -> Potenz, in die die Basis gehoben werden soll. |
| Ergebnis:  | ffp1                                                                                                                                     | -> Ergebnis der Potenzierung.                    |
| Datentyp:  | FLOAT ffp2, ffp3;<br>FLOAT ffp1;                                                                                                         |                                                  |
| Sonstiges: | Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |                                                  |

### 15.2.15 SPSqrt

|            |                                                                                                                                          |                                                      |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|
| Syntax:    | ffp1 = SPSqrt(ffp2);                                                                                                                     |                                                      |
| Funktion:  | Ermittelt die Quadratwurzel einer FFP-Zahl.                                                                                              |                                                      |
| Parameter: | ffp2                                                                                                                                     | -> FFP-Zahl, aus der die Wurzel gezogen werden soll. |
| Ergebnis:  | ffp1                                                                                                                                     | -> Quadratwurzel der FFP-Zahl.                       |
| Datentyp:  | FLOAT ffp2;<br>FLOAT ffp1;                                                                                                               |                                                      |
| Sonstiges: | Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |                                                      |

### 15.2.16 SPTieee

|            |                                                                                                                                          |                                         |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------|
| Syntax:    | int = SPTieee(ffp);                                                                                                                      |                                         |
| Funktion:  | Wandelt eine FFP-Zahl in eine IEEE-Zahl um.                                                                                              |                                         |
| Parameter: | ffp                                                                                                                                      | -> FFP-Zahl, die gewandelt werden soll. |
| Ergebnis:  | int                                                                                                                                      | -> IEEE-Zahl.                           |
| Datentyp:  | FLOAT ffp;<br>int int;                                                                                                                   |                                         |
| Sonstiges: | Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15. |                                         |
| Referenz:  | Siehe auch SPFieee                                                                                                                       |                                         |

### 15.2.17 SPFieee

|            |                                             |                                            |
|------------|---------------------------------------------|--------------------------------------------|
| Syntax:    | ffp = SPFieee(int);                         |                                            |
| Funktion:  | Wandelt eine IEEE-Zahl in eine FFP-Zahl um. |                                            |
| Parameter: | int                                         | -> IEEE-Zahl, die umgewandelt werden soll. |
| Ergebnis:  | ffp                                         | -> FFP-Zahl.                               |
| Datentyp:  | int int;<br>FLOAT ffp;                      |                                            |

**Sonstiges:** Es ist zu beachten, daß die FFP-Zahl bei dem Lattice-C-Compiler in einer anderen Weise bereitgestellt werden muß. Siehe dazu Kapitel 15.

**Referenz:** Siehe auch SPTieec

```
1 /*****
2
3 Mathffp-Demonstration
4 last update 26/05/87
5 von Joerg Koch und Frank Kremser
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Fuehrt alle Befehle aus, die die Mathematik-Bibliothek `Mathtrans
11 enthaelt.
12
13 *****/
14
15 #include <exec/types.h> /* Include-Files laden */
16 #include <exec/tasks.h>
17 #include <exec/libraries.h>
18 #include <exec/devices.h>
19 #include <devices/keymap.h>
20 #include <graphics/copper.h>
21 #include <graphics/display.h>
22 #include <graphics/gfxbase.h>
23 #include <graphics/text.h>
24 #include <graphics/view.h>
25 #include <graphics/gels.h>
26 #include <graphics/regions.h>
27 #include <graphics/sprite.h>
28 #include <hardware/blit.h>
29 #include <intuition/intuition.h>
30 #include <intuition/intuitionbase.h>
31 #include <math.h>
32
33 extern int SPAtan(); /* Nur fuer Lattice-C Benutzer notwendig */
34 extern int SPSin();
35 extern int SPCos();
36 extern int SPTan();
37 extern int SPSincos();
38 extern int SPSinh();
39 extern int SPCosh();
40 extern int SPTanh();
41 extern int SPExp();
42 extern int SPLog();
43 extern int SPPow();
44 extern int SPSqrt();
45 extern int SPTieec();
46 extern int SPFieec();
47
48 int MathBase; /* Lib - Zeiger */
49 int MathTransBase;
50
51 ergebnis(zahl) /* Ergebnis in Float-Zahl wandeln und */
52 int zahl; /* ausgeben */
53 {
54 union test
55 {
```

```
56 FLOAT f;
57 int i;
58 } z;
59
60 z.i = SPTieeee(zahl);
61 printf("%f",z.f);
62 printf("\n");
63 }
64
65 main()
66 {
67 union test1
68 {
69 FLOAT f1;
70 int i1;
71 } zahl1;
72
73 union test2
74 {
75 FLOAT f2;
76 int i2;
77 } zahl2;
78
79 /* Libs oeffnen */
80 if ((MathBase = OpenLibrary("mathffp.library", 0)) == 0) exit();
81
82 if ((MathTransBase = OpenLibrary("mathtrans.library", 0)) == 0) exit();
83
84 zahl1.f1 = 0.2;
85 zahl1.i1 = SPFieeee(zahl1.i1);
86 zahl2.i2 = SPSqrt(zahl1.i1);
87 printf("Wurzel aus 0.2 ist");
88 ergebnis(zahl2.i2);
89
90 zahl1.f1 = 0.2;
91 zahl1.i1 = SPFieeee(zahl1.i1);
92 zahl2.i2 = SPAtan(zahl1.i1);
93 printf("Arcustangens von 0.2 ist");
94 ergebnis(zahl2.i2);
95
96 zahl1.f1 = 0.2;
97 zahl1.i1 = SPFieeee(zahl1.i1);
98 zahl2.i2 = SPSin(zahl1.i1);
99 printf("Sinus von 0.2 ist");
100 ergebnis(zahl2.i2);
101
102 zahl1.f1 = 0.2;
103 zahl1.i1 = SPFieeee(zahl1.i1);
104 zahl2.i2 = SPCos(zahl1.i1);
105 printf("Cosinus von 0.2 ist");
106 ergebnis(zahl2.i2);
107
108 zahl1.f1 = 0.2;
109 zahl1.i1 = SPFieeee(zahl1.i1);
110 zahl2.i2 = SPSinh(zahl1.i1);
111 printf("Hyperbolik-Sinus von 0.2 ist");
112 ergebnis(zahl2.i2);
113
114 zahl1.f1 = 0.2;
115 zahl1.i1 = SPFieeee(zahl1.i1);
116 zahl2.i2 = SPCosh(zahl1.i1);
117 printf("Hyperbolik-Cosinus von 0.2 ist");
118 ergebnis(zahl2.i2);
```

```
119
120 zahl1.f1 = 0.2;
121 zahl1.i1 = SPFieeee(zahl1.i1);
122 zahl2.i2 = SPTanh(zahl1.i1);
123 printf("Hyperbolik-Tangens von 0.2 ist");
124 ergebnis(zahl2.i2);
125
126 zahl1.f1 = 0.2;
127 zahl1.i1 = SPFieeee(zahl1.i1);
128 zahl2.i2 = SPExp(zahl1.i1);
129 printf("e hoch 0.2 ist");
130 ergebnis(zahl2.i2);
131
132 zahl1.f1 = 0.2;
133 zahl1.i1 = SPFieeee(zahl1.i1);
134 zahl2.i2 = SPLog(zahl1.i1);
135 printf("nat. Logarithmus von 0.2 ist");
136 ergebnis(zahl2.i2);
137
138 zahl1.f1 = 0.2;
139 zahl1.i1 = SPFieeee(zahl1.i1);
140 zahl2.i2 = SPPow(zahl1.i1,zahl1.i1);
141 printf("0.2 hoch 0.2 ist");
142 ergebnis(zahl2.i2);
143
144 CloseLibrary(MathTransBase); /* Libs schliessen */
145 CloseLibrary(MathBase);
146 }
```

## Das IFF-Bild-Format

Bei Rechnern früherer Generationen bestand oft die Schwierigkeit, Daten von einem Rechner zu einem anderen Rechner zu übertragen, denn die Bilddaten waren oftmals nicht kompatibel zueinander. Zeichenprogramme konnten oft das Format eines anderen Zeichenprogramms nicht lesen. Mit der Einführung des Amiga sollte diesem Chaos endlich ein Ende gemacht werden. Electronic Arts und Commodore-Amiga entwickelten ein Standardformat, das IFF-Interchange-File-Format. Mit diesem Standard ist der Austausch von standardisierten Daten ohne Schwierigkeiten möglich. In diesem Kapitel wollen wir jedoch nur auf den IFF-Bildstandard eingehen.

Commodore-Amiga empfiehlt jedem Programmierer, sich an diesen Standard zu halten. Möchte man problemlos Bilder, die mit DeluxePaint, Graphicraft oder einem anderen Malprogramm erstellt wurden, in eigenen Programmen verwenden, so muß man sich diesem Standard anpassen oder die fertigen Bilder konvertieren und in einem eigenen Format abspeichern, wie wir es in Kapitel 10 »Ein- und Ausgabe« praktiziert haben. Dies ist sehr umständlich, da z.B. unserer Konvertierungsprogramm in Basic geschrieben ist und getrennt geladen werden muß. Also bleibt einem nichts anderes übrig, als mit der IFF-Welle zu schwimmen.

Der Aufbau dieser IFF-Bild-Files scheint zunächst sehr komplex zu sein, wenn Sie mal unsere IFF-Read- und IFF-Write-Listings betrachten. Zum IFF-Read-Programm sei an dieser Stelle gesagt, daß keine DeluxePaint-Bilder direkt geladen werden können, da diese zusätzlich komprimiert sind. Diese Bilder müssen zunächst mit Graphicraft geladen und abgespeichert werden. Umgekehrt, beim Abspeichern der Bild-Daten im IFF-Format mit dem IFF-Write-Programm bestehen keine Schwierigkeiten, da diese Daten sowohl mit DeluxePaint, als auch, wenn das Bildformat nicht größer als 320 x 200 ist, mit Graphicraft geladen und weiterverarbeitet werden können. Die Daten, um fließende Farbanimation darzustellen, haben wir beim Speichern und Lesen nicht berücksichtigt.

Das Standard-Format besteht aus einzelnen Teilen, die durch einen 4-stelligen Titel angezeigt werden. Nach jedem Titel folgt eine bestimmte Anzahl Daten, woraus der Programmier z.B. die Größe der BitMaps erkennen kann. Die Daten innerhalb dieser Teile haben eine festgelegte Reihenfolge, wodurch das Auffinden von bestimmten Parametern stark vereinfacht wird. Die Titel, die ein IFF-Bild enthalten kann, sind:

Zu Beginn stehen folgende Einträge:

|       |                                                            |
|-------|------------------------------------------------------------|
| FORM- | Kopf des IFF-Files, enthält die Länge der Datei.           |
| ILBM- | Kopf, der den Beginn der nachfolgenden Daten kennzeichnet. |

Im Anschluß daran stehen die folgenden Daten in der Datei. Die ist aber nicht zwangsläufig so. Da nicht alle Daten unbedingt mitgespeichert werden müssen, können einige fehlen oder aber mehrmals auftreten. Zudem besteht keine festgelegte Reihenfolge. Allein »BODY« und die zugehörigen Bild-daten müssen am Ende der Datei stehen.

|                      |                                              |
|----------------------|----------------------------------------------|
| DPPV -               | Graphicraft-spezifisch.                      |
| BMHD- (BitMapHeader) | Grafikinformation.                           |
| CMHD-                | Graphicraft-spezifisch.                      |
| CMAP- (ColorMap)     | Farb informationen.                          |
| CRNG- (ColorRange)   | Informationen zu Farbverläufen.              |
| BODY-                | Bilddaten der Grafik.                        |
| CAMG-                | Amiga-spezifisch: ViewPort-Mode des Screens. |
| CCRT- (CycleInfo)    | Informationen zur Farbanimation.             |

Wie schon erwähnt, folgen diesen Titeln Daten in einer gewissen Reihenfolge, so daß man nicht alle Titel abfragen muß, um an bestimmte Parameter zu gelangen. Wichtig sind CMAP, BODY, BMHD und FORM, da sie die Grundinformationen der Grafik enthalten:

Kopf:

|             |                           |
|-------------|---------------------------|
| vier Bytes: | "FORM"                    |
| vier Bytes: | Größe der Datei in Bytes. |
| vier Bytes: | "ILBM"                    |

---

BitMapHeader :

|             |                                       |
|-------------|---------------------------------------|
| vier Bytes: | "BMHD"                                |
| vier Bytes: | Wert 20 als LONGCARD (Länge von BMHD) |
| zwei Bytes: | Breite des Screens                    |
| zwei Bytes: | Höhe des Screens                      |
| vier Bytes: | Wert 0                                |
| ein Byte:   | Tiefe des Screens                     |
| fünf Bytes: | Wert 0                                |
| ein Byte:   | Wert 10                               |

|             |                    |
|-------------|--------------------|
| ein Byte:   | Wert 11            |
| zwei Bytes: | Breite des Screens |
| zwei Bytes: | Höhe des Screens   |

---

**ColorMap:**

|               |                                                                         |
|---------------|-------------------------------------------------------------------------|
| vier Bytes:   | "CMAP"                                                                  |
| vier Bytes:   | Anzahl der Farben mal 3                                                 |
| anschließend: | für jede Farbe je 3 Bytes (Rot, Grün, Blau werden getrennt gespeichert) |

---

**Body :**

|               |                                                                                                                    |
|---------------|--------------------------------------------------------------------------------------------------------------------|
| vier Bytes:   | "BODY"                                                                                                             |
| vier Bytes:   | $\text{Größe der Grafik} = \text{Höhe} * \text{Breite} / 8 * \text{Tiefe}$                                         |
| anschließend: | Grafik, reihenweise gespeichert. Also: erste Zeile von erster BitPlane, dann erste Zeile von zweiter BitPlane usw. |

Die Art dieses Aufbaus können Sie auch erkennen, wenn Sie ein Bildfile mit »type opt h« vom CLI aus listen. Diese Grundstruktur wird sowohl beim Laden als auch beim Speichern von IFF-Bild-Files verwendet.

```
1 /*****
2
3 IFF-Read-Demonstration
4 last update 26/05/87
5 von Frank Kremser und Joerg Koch
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Diese laedt ein Bild im IFF-Format ein. Der Screen passt sich automatisch
11 dem Bildformat an. DPaint Bilder muessen erst Graphicraft konvertiert
12 werden, da diese Bilder komprimiert sind.
13
14 *****/
15
16 #include <exec/types.h> /* Include-Files laden */
17 #include <exec/tasks.h>
18 #include <exec/libraries.h>
19 #include <exec/devices.h>
20 #include <exec/memory.h>
21 #include <devices/keymap.h>
22 #include <graphics/copper.h>
23 #include <graphics/display.h>
24 #include <graphics/gfxbase.h>
25 #include <graphics/text.h>
26 #include <graphics/view.h>
27 #include <graphics/gels.h>
28 #include <graphics/regions.h>
29 #include <graphics/sprite.h>
30 #include <hardware/blit.h>
31 #include <libraries/dos.h>
32 #include <intuition/intuition.h>
33 #include <intuition/intuitionbase.h>
34
35
36 struct IntuitionBase *IntuitionBase; /* Lib - Zeiger */
37 struct GfxBase *GfxBase;
38 ULONG DosBase;
39
40 BYTE *puffer[5]; /* max 5 BitPlane-Zeiger */
41
42 char dppv[] = {"DPPV"}; /* IFF Titel */
43 char bmhd[] = {"BMHD"};
44 char cmap[] = {"CMAP"};
45 char crng[] = {"CRNG"};
46 char body[] = {"BODY"};
47 char camg[] = {"CAMG"};
48 char ccrt[] = {"CCRT"};
49
50 struct Screen *screen;
51
52 struct NewScreen ns = /* Screen definieren */
53 { /* wird spaeter konfiguriert */
54 0,
55 0,
56 0,
57 0,
58 0,
59 0,
60 1,
61 NULL,
62 CUSTOMSCREEN,
63 NULL,
64 NULL,
```

```

65 NULL,
66 NULL
67 };
68
69
70 main()
71 {
72 LONG schleife, schleif, fanz;
73 ULONG datei, mull;
74 BYTE rot, grun, blau;
75 char text[3];
76 BOOL ende;
77 struct RastPort *rp;
78 struct BitMap *ptr;
79
80
81 if ((GfxBase = (struct GfxBase *)
82 OpenLibrary("graphics.library", 0)) == 0) exit();
83
84 if ((IntuitionBase = (struct IntuitionBase *)
85 OpenLibrary("intuition.library", 0)) == 0) exit();
86 } /* Libs oeffnen */
87 if ((DosBase =
88 OpenLibrary("dos.library", 0)) == 0) exit();
89
90 datei = Open("DFO:IFFBILD",MODE_OLDFILE); /* Datei oeffnen */
91 if(datei == 0) /* Filename: Iffbild */
92 {
93 printf("Keinen File geoeffnet !!!\n");
94 exit();
95 };
96
97 mull = AllocMem(25,MEMF_CHIP;MEMF_CLEAR); /* Muell-Speicher */
98 ende = FALSE;
99 Read(datei,mull,12); /* Header laden */
100
101 while(ende == FALSE)
102 {
103 Read(datei,&text[0],4);
104
105 if(strncmp(&text[0],&bmhd[0],4) == 0) /* BMHD laden */
106 {
107 Read(datei,mull,4);
108 Read(datei,&ns.Width,2);
109 Read(datei,&ns.Height,2);
110 Read(datei,mull,3);
111 Read(datei,&ns.Depth,2);
112 Read(datei,mull,11);
113 if(ns.Width > 320) ns.ViewModes := HIRRES;
114 if(ns.Height > 256) ns.ViewModes := INTERLACE;
115
116 /* Screen oeffnen */
117 if ((screen = (struct Screen*) OpenScreen(&ns)) == NULL) exit();
118
119 rp = &screen->RastPort; /* BitMapPointer bestimmen */
120 ptr = rp->BitMap;
121 for(schleife = 0; schleife < ns.Depth; schleife++)
122 puffer[schleife] = ptr->Planes[schleife];
123 };
124
125 /* CMAP laden */
126 if(strncmp(&text[0],&cmap[0],4) == 0)
127 {
128 Read(datei,&fanz,4);
129 fanz = fanz / 3;

```

```

129 for(schleife = 0; schleife < fanz; schleife++)
130 {
131 Read(datei,&rot,1);
132 Read(datei,&grun,1);
133 Read(datei,&blau,1);
134 SetRGBA(&screen->ViewPort,schleife,rot/16,grun/16,blau/16);
135 };
136 };
137 /* BODY laden */
138 if(strncmp(&text[0],&body[0],4) == 0)
139 {
140 Read(datei,mull,4);
141 for(schleif = 0; schleif < ns.Height; schleif++)
142 for(schleife = 0; schleife < ns.Depth; schleife++)
143 Read(datei,puffer[schleife] + (ns.Width / 8 * schleif),
144 ns.Width / 8);
145 ende = TRUE;
146 };
147
148 if(strncmp(&text[0],&crmg[0],4) == 0) /* Zusatzinformationen */
149 Read(datei,mull,12); /* uebergehen, falls */
150 /* vorhanden */
151 if(strncmp(&text[0],&dppv[0],4) == 0)
152 Read(datei,mull,108);
153
154 if(strncmp(&text[0],&camg[0],4) == 0)
155 Read(datei,mull,8);
156
157 if(strncmp(&text[0],&ccrt[0],4) == 0)
158 Read(datei,mull,18);
159 };
160
161 Close(datei); /* Datei schliessen */
162
163 for(schleife = 0; schleife < 750000; ++schleife);
164
165 CloseScreen(screen); /* Screen und Libs */
166 CloseLibrary(DosBase); /* schliessen */
167 CloseLibrary(IntuitionBase);
168 CloseLibrary(GfxBase);
169 }

1 /*****
2
3 IFF-Write-Demonstration
4 last update 26/05/87
5 von Joerg Koch und Frank Kremser
6 (c) Markt & Technik 1987
7
8 *****/
9
10 Dieses Programm oeffnet einen Screen und speichert ihn als IFF-
11 File ab.
12
13 *****/
14
15 #include <exec/types.h> /* Include-Files laden */
16 #include <exec/tasks.h>
17 #include <exec/libraries.h>
18 #include <exec/devices.h>
19 #include <exec/memory.h>
20 #include <devices/keymap.h>

```

```

21 #include <graphics/copper.h>
22 #include <graphics/display.h>
23 #include <graphics/gfxbase.h>
24 #include <graphics/text.h>
25 #include <graphics/view.h>
26 #include <graphics/gels.h>
27 #include <graphics/regions.h>
28 #include <graphics/sprite.h>
29 #include <hardware/blit.h>
30 #include <libraries/dos.h>
31 #include <intuition/intuition.h>
32 #include <intuition/intuitionbase.h>
33
34
35 struct IntuitionBase *IntuitionBase; /* Lib - Zeiger */
36 struct GfxBase *GfxBase;
37 ULONG DosBase;
38
39 BYTE *puffer[5]; /* max 5 Puffer-Zeiger */
40
41 char form[] = {"FORM"}; /* IFF-Titel */
42 char ilbm[] = {"ILBM"};
43 char dppv[] = {"DPPV"};
44 char bmhd[] = {"BMHD"};
45 char cmap[] = {"CMAP"};
46 char crng[] = {"CRNG"};
47 char body[] = {"BODY"};
48 char camg[] = {"CAMG"};
49 char ccrt[] = {"CCRT"};
50
51 struct Screen *screen;
52
53 struct NewScreen ns = /* Screen definieren */
54 {
55 0,
56 0,
57 320,
58 256,
59 5,
60 0,
61 1,
62 NULL,
63 CUSTOMSCREEN,
64 NULL,
65 NULL,
66 NULL,
67 NULL
68 };
69
70
71 main()
72 {
73 LONG schleife, schleif, fanz;
74 ULONG datei, dat;
75 WORD farbe;
76 BYTE rot, grun, blau, dati;
77 struct RastPort *rp;
78 struct BitMap *ptr;
79
80
81 if ((GfxBase = (struct GfxBase *)
82 OpenLibrary("graphics.library", 0)) == 0) exit();
83

```

```
84 if ((IntuitionBase = (struct IntuitionBase *)
85 OpenLibrary("intuition.library", 0)) == 0) exit();
86 /* Libs oeffnen */
87 if ((DosBase =
88 OpenLibrary("dos.library", 0)) == 0) exit();
89
90 datei = Open("DFO:DEMOSCREEN",MODE_NEWFILE); /* Datei oeffnen */
91 if(datei == 0) /* IFF-File wird */
92 { /* als */
93 printf("Keinen File geoeffnet !!!\n"); /* Demoscreen */
94 exit(); /* abgespeichert */
95 };
96 /* Screen oeffnen */
97 if ((screen = (struct Screen*) OpenScreen(&ns)) == NULL) exit();
98
99 fanz = 1;
100 rp = &screen->RastPort; /* BitPlane-Zeiger ermitteln */
101 ptr = rp->BitMap;
102 for(schleife = 0; schleife < ns.Depth; schleife++)
103 {
104 puffer[schleife] = ptr->Planes[schleife];
105 fanz = fanz * 2;
106 };
107 /* Header schreiben */
108 Write(datei,&form[0],4);
109 dat = 48 + (fanz * 3) + (ns.Width / 8 * ns.Height * ns.Depth);
110 Write(datei,&dat,4);
111 Write(datei,&ilbm[0],4);
112
113 Write(datei,&bmhd[0],4); /* BMHD schreiben */
114 dat = 20;
115 Write(datei,&dat,4);
116 Write(datei,&ns.Width,2);
117 Write(datei,&ns.Height,2);
118 dat = 0;
119 Write(datei,&dat,4);
120 dati = (BYTE) ns.Depth;
121 Write(datei,&dati,1);
122 Write(datei,&dat,1);
123 Write(datei,&dat,4);
124 dati = 10;
125 Write(datei,&dati,1);
126 dati = 11;
127 Write(datei,&dati,1);
128 Write(datei,&ns.Width,2);
129 Write(datei,&ns.Height,2);
130
131 Write(datei,&cmap[0],4); /* CMAP schreiben */
132 fanz = fanz * 3;
133 Write(datei,&fanz,4);
134 fanz = fanz / 3;
135 for(schleife = 0; schleife < fanz; schleife++)
136 {
137 farbe = GetRGB4(screen->ViewPort.ColorMap,schleife);
138 rot = farbe / 256;
139 rot = rot * 16;
140 farbe = farbe - (rot * 16); /* Farben berechnen */
141 grun = farbe / 16;
142 grun = grun * 16;
143 farbe = farbe - grun;
144 blau = farbe * 16;
145 Write(datei,&rot,1);
146 Write(datei,&grun,1);
147 Write(datei,&blau,1);
```

```
148);
149 /* BODY schreiben */
150 Write(datei,&body[0],4);
151 dat = ns.Width / 8 * ns.Height * ns.Depth;
152 Write(datei,&dat,4);
153 for(schleif = 0; schleif < ns.Height; schleif++)
154 for(schleife = 0; schleife < ns.Depth; schleife++)
155 Write(datei,puffer[schleife] + (ns.Width / 8 * schleif),
156 ns.Width / 8);
157
158 Close(datei); /* File schliessen */
159
160 for(schleife = 0; schleife < 20000; ++schleife);
161
162 CloseScreen(screen); /* Libs und Screen schliessen */
163 CloseLibrary(DosBase);
164 CloseLibrary(IntuitionBase);
165 CloseLibrary(GfxBase);
166)
```



## Sonstige Befehle

In diesem Kapitel werden Befehle erläutert, die nicht in die vorherigen Kapitel eingeordnet werden konnten, aber trotzdem wichtige Funktionen erfüllen.

Da diese Befehle allerdings zu unterschiedlich sind, kann für dieses Kapitel ausnahmsweise kein Demonstrationsprogramm angeführt werden. Da die Befehle aber recht einfache Funktionen erfüllen, glauben wir, daß dies kein allzugroßer Nachteil ist.

## 17.1 CurrentTime

- Syntax:** CurrentTime(&Sekunden,&Mikrosek);
- Funktion:** Holt die Systemzeit.
- Parameter:** Sekunden -> Langwort-Variable, in der die Sekunden der Systemzeit gespeichert werden  
Mikrosek -> Langwort-Variable, in der die Mikrosekunden gespeichert werden
- Ergebnis:** Kein Ergebnis.
- Datentyp:** LONG Sekunden, Mikrosek;
- Sonstiges:** Stunden und Minuten der Systemzeit müssen aus der Sekundenzeit ermittelt werden. Da die Sekundenvariable 4 Bytes (Langwort) lang ist, kann sie Werte bis zu  $2^{32}$  Sekunden, also etwa 139 Jahre, aufnehmen.
- Die Sekunden- und Mikrosekunden-Variablen werden folgendermaßen deklariert:  
LONG Sekunden, Mikrosek;

## 17.2 DoubleClick

|            |                                                                                                                            |
|------------|----------------------------------------------------------------------------------------------------------------------------|
| Syntax:    | gut = DoubleClick(SSekunden,SMikro,ESekunden,EMikro);                                                                      |
| Funktion:  | Prüft, ob ein Maus-Doppelklick in Ordnung war                                                                              |
| Parameter: | SSekunden,<br>SMikro           -> Startzeit des Doppelklicks<br>ESekunden,<br>EMikro           -> Endzeit des Doppelklicks |
| Ergebnis:  | gut               -> Ist TRUE, falls er in Ordnung war,<br>FALSE, falls nicht                                              |
| Datentyp:  | LONG SSekunden, SMikro, ESekunden, EMikro;                                                                                 |
| Sonstiges: | Die Start- und Endzeiten können mittels CurrentTime ermittelt werden.                                                      |

Anschließend vergleicht dieser Befehl die benötigte Zeit mit der erlaubten Zeit, die mit Preference festgelegt ist. War sie zu lang, wird FALSE zurückgegeben, das heißt, daß dies kein erlaubter Doppelklick war. War er erlaubt, wird TRUE zurückgegeben.



# Anhang A

## Zuweisungen Befehle <-> Libraries

Aus der folgenden Liste kann abgelesen werden, welche Libraries mit »#include« einzulesen sind, um bestimmte Befehle verwenden zu können:

|               |              |
|---------------|--------------|
| clist.library | AllocCList ) |
|               | ConcatCList  |
|               | CopyCList )  |
|               | FlushCList   |
|               | FreeCList    |
|               | GetCLBuf     |
|               | GetCLChar    |
|               | GetCLWord    |
|               | IncrCLMark   |
|               | InitCLPool   |
|               | MarkCList    |
|               | PeekCLMark   |
|               | PutCLBuf     |
|               | PutCLChar    |
|               | PutCLWord    |
|               | SizeCList    |
|               | SplitCList   |
|               | SubCList     |
|               | UnGetCLChar  |
|               | UnGetCLWord  |
|               | UnPutCLChar  |
|               | UnPutCLWord  |

|                  |               |
|------------------|---------------|
| dos.library      | Close         |
|                  | CreateDir     |
|                  | CreateProc    |
|                  | CurrentDir    |
|                  | DateStamp     |
|                  | Delay         |
|                  | DeleteFile    |
|                  | DeviceProc    |
|                  | DupLock       |
|                  | Examine       |
|                  | Execute       |
|                  | Exit          |
|                  | ExNext        |
|                  | Info          |
|                  | Input         |
|                  | IoErr         |
|                  | IsInteractive |
|                  | LoadSeg       |
|                  | Lock          |
|                  | Open          |
|                  | Output        |
|                  | ParentDir     |
|                  | Read          |
|                  | Rename        |
|                  | Seek          |
|                  | SetComment    |
|                  | SetProtection |
|                  | UnLoadSeg     |
|                  | UnLock        |
|                  | WaitForChar   |
|                  | Write         |
| diskfont.library | AvailFonts    |
|                  | OpenDiskFont  |

---

|              |              |
|--------------|--------------|
| exec.library | AddDevice    |
|              | AddHead      |
|              | AddIntServer |
|              | AddLibrary   |
|              | AddPort      |
|              | AddResource  |
|              | AddTail      |
|              | AddTask      |
|              | Alert        |
|              | Allocat      |
|              | AllocEntry   |
|              | AllocMem     |
|              | AllocSignal  |
|              | AllocTrap    |
|              | AvailMem     |
|              | Cause        |
|              | CheckIO      |
|              | CloseDevice  |
|              | CloseLibrary |
|              | ColdReset    |
|              | Deallocate   |
|              | Disable      |
|              | DoIO         |
|              | Enable       |
|              | Enqueue      |
|              | FindName     |
|              | FindPort     |
|              | FindTask     |
|              | Forbid       |
|              | FreeEntry    |
|              | FreeMem      |
|              | FreeSignal   |
|              | FreeTrap     |
|              | GetCC        |
|              | GetMsg       |
|              | InitStruct   |
|              | Insert       |
|              | MakeLibrary  |
|              | OpenDevice   |
|              | OpenLibrary  |
|              | OpenResource |
|              | Permit       |
|              | PutMsg       |

|                      |                 |
|----------------------|-----------------|
|                      | RemDevice       |
|                      | RemHead         |
|                      | RemIntServer    |
|                      | RemLibrary      |
|                      | Remove          |
|                      | RemPort         |
|                      | RemResource     |
|                      | RemTail         |
|                      | RemTask         |
|                      | ReplyMsg        |
|                      | SendIO          |
|                      | SetExcept       |
|                      | SetFunction     |
|                      | SetIntVector    |
|                      | SetSignal       |
|                      | SetSR           |
|                      | SetTaskPri      |
|                      | Signal          |
|                      | SumLibrary      |
|                      | SuperState      |
|                      | UserState       |
|                      | Wait            |
|                      | WaitIO          |
|                      | WaitPort        |
| exec_support.library | CreateExtIO     |
|                      | CreateStdIO     |
|                      | DeletePort      |
|                      | DeleteStdIO     |
| graphics.library     | AddAnimOb       |
|                      | AddBob          |
|                      | AddFont         |
|                      | AddVSprite      |
|                      | AllocRaster     |
|                      | AndRectRegion   |
|                      | AndRegionRegion |
|                      | Animate         |
|                      | AreaDraw        |
|                      | AreaEnd         |
|                      | AreaMove        |
|                      | AskFont         |
|                      | AskSoftStyle    |
|                      | BlitBitMap      |

BltBitMapRastPort  
BltClear  
BltPattern  
BltTemplate  
CEND  
ChangeSprite  
CINIT  
ClearEOL  
ClearRegion  
ClearScreen  
ClipBlit  
CloseFont  
CMOVE  
CopySBitMap  
CWAIT  
DisownBlitter  
DisposeRegion  
DoCollision  
Draw  
DrawGLList  
Flood  
FreeColorMap  
FreeCopList  
FreeCprList  
FreeGBuffers  
FreeRaster  
FreeSprite  
FreeVPortCopLists  
GetColorMap  
GetGBuffers  
GetRGB4  
GetSprite  
InitArea  
InitBitMap  
InitGels  
InitGMasks  
InitMasks  
InitRastPort  
InitTmpRas  
InitView  
InitVPort  
LoadRGB4  
LoadView

LockLayerRom  
MakeVPort  
Move  
MoveSprite  
MrgCop  
NewRegion  
OpenFont  
OrRectRegion  
OrRegionRegion  
OwnBlitter  
PolyDraw  
QBlit  
QBSBlit  
ReadPixel  
RectFill  
RemFont  
RemIBob  
RemVSprite  
ScrollRaster  
ScrollVPort  
SetAPen  
SetBPen  
SetOPen  
SetCollision  
SetDrMd  
SetFont  
SetRast  
SetRGB4  
SetSoftStyle  
SortGList  
SyncSBitMap  
Text  
TextLength  
UnlockLayerRom  
VBeamPos  
WaitBlit  
WaitBOVP  
WaitTOF  
WritePixel  
XorRectRegion  
XorRegionRegion  
icon AddFreeList  
AllocWBOject

---

|                   |                 |
|-------------------|-----------------|
|                   | BumpRevision    |
|                   | FindToolType    |
|                   | FreeDiskObject  |
|                   | FreeFreeList    |
|                   | FreeWBOBJECT    |
|                   | GetDiskObject   |
|                   | GetIcon         |
|                   | GetWBOBJECT     |
|                   | MatchToolValue  |
|                   | PutDiskObject   |
|                   | PutIcon         |
|                   | PutWBOBJECT     |
| intuition.library | AddGadget       |
|                   | AllocRemember   |
|                   | AutoRequest     |
|                   | BeginRefresh    |
|                   | BuildSysRequest |
|                   | ClearDMRequest  |
|                   | ClearMenuStrip  |
|                   | ClearPointer    |
|                   | CloseScreen     |
|                   | CloseWindow     |
|                   | CloseWorkBench  |
|                   | CurrentTime     |
|                   | DisplayAlert    |
|                   | DisplayBeep     |
|                   | DoubleClick     |
|                   | DrawBorder      |
|                   | DrawImage       |
|                   | EndRefresh      |
|                   | EndRequest      |
|                   | FreeRemember    |
|                   | FreeSysRequest  |
|                   | GetDefPrefs     |
|                   | GetPrefs        |
|                   | InitRequester   |
|                   | IntuiTextLength |
|                   | ItemAddress     |
|                   | MakeScreen      |
|                   | ModifyIDCMP     |
|                   | ModifyProp      |
|                   | MoveScreen      |
|                   | MoveWindow      |

OffGadget  
OffMenu  
OnGadget  
OnMenu  
OpenScreen  
OpenWindow  
OpenWorkBench  
PrintIText  
RefreshGadgets  
RemakeDisplay  
RemoveGadget  
ReportMouse  
Request  
RethinkDisplay  
ScreenToBack  
ScreenToFront  
SetDMRequest  
SetMenuStrip  
SetPointer  
SetPrefs  
SetWindowTitels  
ShowTitle  
SizeWindow  
ViewAddress  
ViewPortAddress  
WBenchToBack  
WBenchToFront  
WindowLimits  
WindowToBack  
WindowToFront

layers.library

BeginUpdate  
BehindLayer  
CreateBehindLayer  
CreateUpfrontLayer  
DeleteLayer  
DisposeLayerInfo  
EndUpdate  
FattenLayerInfo  
InitLayers  
LockLayer  
LockLayerInfo  
LockLayers  
MoveLayer

---

|                   |                          |
|-------------------|--------------------------|
|                   | MoveLayerInFrontOf       |
|                   | NewLayerInfo             |
|                   | ScrollLayer              |
|                   | SizeLayer                |
|                   | SwapBitsRastPortClipRect |
|                   | ThinLayerInfo            |
|                   | UnlockLayer              |
|                   | UnlockLayerInfo          |
|                   | UnlockLayers             |
|                   | UpfrontLayer             |
|                   | WhichLayer               |
| mathffp.library   | abs                      |
|                   | faddi                    |
|                   | fcmpli                   |
|                   | fdivi                    |
|                   | fflti                    |
|                   | fmuli                    |
|                   | fnegi                    |
|                   | fsubi                    |
|                   | ftsti                    |
|                   | SPAbs                    |
|                   | SPAdd                    |
|                   | SPCmp                    |
|                   | SPDiv                    |
|                   | SPFlt                    |
|                   | SPMul                    |
|                   | SPNeg                    |
|                   | SPSub                    |
|                   | SPTst                    |
| mathtrans.library | SPAcos                   |
|                   | APAsin                   |
|                   | APAtan                   |
|                   | SPCos                    |
|                   | SPCosh                   |
|                   | SPExp                    |
|                   | SPFicce                  |
|                   | SPLog                    |
|                   | SPLog10                  |
|                   | SPPow                    |
|                   | SPSin                    |
|                   | SPSincos                 |
|                   | SPSinh                   |

SPSqrt  
SPTan  
SPTanh  
SPTieee

mathieedoubbas.library IEEEEDPAbs  
IEEEEDPAdd  
IEEEEDPCmp  
IEEEEDPDiv  
IEEEEDPFlt  
IEEEEDPMul  
IEEEEDPNeg  
IEEEEDPSub  
IEEEEDPTst

Nicht alle Befehle lassen sich in die obenstehenden Libraries einordnen. Ein Beispiel hierfür sind die Macros, wie OFF\_SPRITE, ON\_SPRITE, OFF\_DISPLAY und ON\_DISPLAY. Sie befinden sich in einer Include-Datei auf der C-Diskette. Soll solch ein Macro verwendet werden, so muß folgender Include-Befehl gegeben werden:

```
#include <graphics/gfxmacros.h>c
```

# Anhang B

## Die Structures

Für die Programmierung des Amiga stehen noch weitaus mehr Structures bereit, als in den Kapiteln beschrieben werden konnte. Aus diesem Grund führen wir an dieser Stelle alle weiteren Structures an.

Wie auf die einzelnen Einträge zugegriffen werden kann, wollen wir anhand eines kurzen Beispiels darstellen:

Nehmen wir an, es gäbe folgende Structure:

```
struct Demo
{
 USHORT eintrag; /* Integer-Eintrag */
 struct Demo amiga; /* Dieser Eintrag besteht aus einer
 weiteren Structure vom Typ Demo */
};
```

Dann könnte sie folgendermaßen deklariert werden:

```
struct Demo test = oder struct Demo test =
{ {
 234, 765,
 NULL viel
}; };
```

Die zweite Möglichkeit ist nur erlaubt, wenn zuvor eine Variable "viel" vom Typ Demo – also dem Structure-Typ – deklariert wurde. Sollen der Structure erst im Programm Werte übergeben werden, so genügt folgende Deklaration:

```
struct Demo test;
```

Sollen der Variablen »test« im Programm nun bestimmte Werte zugewiesen werden oder soll auf Eintragungen zugegriffen werden, so muß das folgendermaßen geschehen:

```
test.eintrag
test.amiga, bzw. test.amiga.eintrag, bzw. test.amiga.amiga usw.
```

Ist "test" als Zeiger deklariert worden, also

```
struct Demo *test;
```

so muß mit

```
test->eintrag
test->amiga usw
```

darauf zugegriffen werden.

Aber nun zu den Structures:

```
struct AnimComp {
 WORD Flags;
 WORD Timer;
 WORD TimeSet;
 struct AnimComp *NextComp;
 struct AnimComp *PrevComp;
 struct AnimComp *NextSeq;
 struct AnimComp *PrevSeq;
 WORD (*AnimCRoutine)();
 WORD YTrans;
 WORD XTrans;
 struct AnimOb *HeadOb;
 struct Bob *AnimBob;
};

struct AnimOb {
 struct AnimOb *NextOb, *PrevOb;
 LONG Clock;
 WORD AnOldY, AnOldX;
 WORD AnY, AnX;
 WORD YVel, XVel;
 WORD YAccel, XAccel;
 WORD RingYTrans, RingXTrans;
 WORD (*AnimORoutine)();
 struct AnimComp *HeadComp;
 AUserStuff AUserExt;
};

struct BlitNode {
 struct BlitNode *n;
 int (*function)();
 char stat;
 short beamsync;
 int (*cleanup)();
};
```

```
struct Bob {
 WORD Flags;
 WORD *SaveBuffer;
 WORD *ImageShadow;
 struct Bob *Before;
 struct Bob *After;
 struct VSprite *BobVSprite;
 struct AnimComp *BobComp;
 struct DBufPacket *DBuffer;
 BUserStuff BUseerExt;
};

struct ClipRect {
 struct ClipRect *Next;
 struct ClipRect *Prev;
 struct Layer *lobs;
 struct BitMap *BitMap;
 struct Rectangle bounds;
 struct ClipRect *_p1, *_p2;
 LONG reserved;
 LONG flags;
};

struct DBufPacket {
 WORD BufY, BufX;
 struct VSprite *BufPath;
 WORD *BufBuffer;
};

struct DiskFontHeader {
 struct Node dfh_DF;
 UWORD dfh_FileID;
 UWORD dfh_Revision;
 LONG dfh_Segment;
 char dfh_Name[MAXFONTNAME];
 struct TextFont dfh_TF;
};

struct DiskObject {
 UWORD do_Magic;
 UWORD do_Version;
 struct Gadget do_Gadget;
 UBYTE do_Type;
 char *do_DefaultTool;
 char **do_ToolTypes;
 LONG do_CurrentX;
 LONG do_CurrentY;
 struct DrawerData *do_DrawerData;
 char *do_ToolWindow;
 LONG do_StackSize;
};
```

```
struct DrawerData {
 struct NewWindow dd_NewWindow;
 LONG dd_CurrentX;
 LONG dd_CurrentY;
 LONG dd_MinX;
 LONG dd_MinY;
 LONG dd_MaxX;
 LONG dd_MaxY;
 struct Gadget dd_HorizScroll;
 struct Gadget dd_VertScroll;
 struct Gadget dd_UpMove;
 struct Gadget dd_DownMove;
 struct Gadget dd_LeftMove;
 struct Gadget dd_RightMove;
 struct Image dd_HorizImage;
 struct Image dd_VertImage;
 struct PropInfo dd_HorizProp;
 struct PropInfo dd_VertProp;
 struct Window *dd_DrawerWin;
 struct WBOject *dd_Object;
 struct List dd_Children;
 LONG dd_Lock;
};

struct FontContents {
 char fc_FileName[MAXFONTPATH];
 UWORD fc_YSize;
 UBYTE fc_Style;
 UBYTE fc_Flags;
};

struct FontContentsHeader {
 UWORD fch_FileID;
 UWORD fch_NumEntries;
};

struct Layer {
 struct Layer *front, *back;
 struct ClipRect *ClipRect;
 struct RastPort *rp;
 struct Rectangle bounds;
 UBYTE Lock;
 UBYTE LockCount;
 UBYTE LayerLockCount;
 UBYTE reserved;
 UWORD reserved1;
 UWORD Flags;
 struct BitMap *SuperBitMap;
 struct ClipRect *SuperClipRect;
 APTR Window;
 SHORT Scroll_X, Scroll_Y;
 struct MsgPort LockPort;
 struct Message LockMessage;
 struct MsgPort ReplyPort;
 struct Message I_LockMessage;
 struct Region *DamageList;
 struct ClipRect *_cliprects;
 struct Layer_Info *Layer_Info;
};
```

```
 struct Task *LayerLocker;
 struct ClipRect *SuperSaveClipRects;
 struct ClipRect *cr, *cr2, *crnew;
 APTR _p1;
};

struct Layer_Info {
 struct Layer *top_layer;
 struct Layer *check_lp;
 struct Layer *obs;
 struct MsgPort RP_ReplyPort;
 struct MsgPort LockPort;
 UBYTE Lock;
 UBYTE broadcast;
 UBYTE LockNest;
 UBYTE Flags;
 struct Task *Locker;
 BYTE fatten_count;
 UBYTE bytereserved;
 UWORD wordreserved;
 UWORD LayerInfo_extra_size;
 ULONG longreserved;
 struct LayerInfo_extra *LayerInfo_extra;
};

struct Library {
 struct Node lib_Node;
 UBYTE lib_Flags;
 UBYTE lib_pad;
 UWORD lib_NegSize;
 UWORD lib_PosSize;
 UWORD lib_Version;
 UWORD lib_Revision;
 APTR lib_IDString;
 ULONG lib_Sum;
 UWORD lib_OpenCnt;
};

struct List {
 struct Node *Ih_Head;
 struct Node *Ih_Tail;
 struct Node *Ih_TailPred;
 UBYTE Ih_Type;
 UBYTE I_pad;
};

struct MemHeader {
 struct Node mh_Node;
 UWORD mh_Attributes;
 struct MemChunk *mh_First;
 APTR mh_Lower;
 APTR mh_Upper;
 ULONG mh_Free;
};
```

```
struct Message {
 struct Node mn_Node;
 struct MsgPort *mn_ReplyPort;
 UWORD mn_Length;
};

struct MsgPort {
 struct Node mp_Node;
 UBYTE mp_Flags;
 UBYTE mp_SigBit;
 struct Task *mp_SigTask;
 struct List mp_MsgList;
};

struct NewScreen {
 SHORT LeftEdge, TopEdge;
 SHORT Width, Height;
 UBYTE DetailPen, BlockPen;
 USHORT ViewModes;
 USHORT Type;
 struct TextAttr *Font;
 UBYTE *DefaultTitle;
 struct Gadget *Gadgets;
 struct BitMap *CustomBitMap;
};

struct NewWindow {
 SHORT LeftEdge, TopEdge;
 SHORT Width, Height;
 UBYTE DetailPen, BlockPen;
 ULONG IDCMPFlags;
 ULONG Flags;
 struct Gadget *FirstGadget;
 struct Image *CheckMark;
 UBYTE *Title;
 struct Screen *Screen;
 struct BitMap *BitMap;
 SHORT MinWidth, MinHeight;
 SHORT MaxWidth, MaxHeight;
 USHORT Type;
};

struct Node {
 struct Node *In_Succ;
 struct Node *In_Pred;
 UBYTE In_Type;
 BYTE In_Pri;
 char *In_Name;
};

struct Preferences {
 BYTE FontHeight;
 UBYTE PrinterPort;
 USHORT BaudRate;
 struct timeval KeyRptSpeed, KeyRptDelay;
 struct timeval DoubleClick;
 USHORT PointerMatrix[POINTERSIZE];
 BYTE XOffset, YOffset;
};
```

```
 USHORT color17, color18, color19;
 USHORT PointerTick;
 USHORT color0, color1, color2, color3;
 BYTE ViewXOffset, ViewYOffset;
 WORD ViewInitX, ViewInitY;
 BOOL EnableCLI;
 USHORT PrinterType;
 BYTE PrinterFilename[FILENAME_SIZE];
 USHORT PrintPitch;
 USHORT PrintQuality;
 USHORT PrintSpacing;
 WORD PrintLeftMargin, PrintRightMargin;
 USHORT PrintImage;
 USHORT PrintAspect;
 USHORT PrintShade;
 WORD PrintThreshold;
 USHORT PaperSize;
 WORD PaperLength;
 USHORT PaperType;
};

struct RastPort {
 struct Layer *Layer;
 struct BitMap *BitMap;
 USHORT *AreaPtrn;
 struct TmpRas *TmpRas;
 struct AreaInfo *AreaInfo;
 struct GelsInfo *GelsInfo;
 BYTE Mask;
 BYTE FgPen;
 BYTE BgPen;
 BYTE AOIPen;
 BYTE DrawMode;
 Byte AreaPtSz;
 Byte linpatcnt;
 BYTE dummy;
 USHORT Flags;
 USHORT LinePtrn;
 SHORT cp_x, cp_y;
 BYTE minterms[8];
 SHORT PenWidth;
 SHORT PenHeight;
 struct TextFont *Font;
 BYTE AlgoStyle;
 BYTE TxFlags;
 WORD TxHeight;
 WORD TxWidth;
 WORD TxBaseline;
 WORD TxSpacing;
 APTR *RP_User;
 WORD wordreserved[7];
 ULONG longreserved[2];
 BYTE reserved[2];
};
```

```
struct Resident {
 UWORD rt_MatchWord;
 struct Resident *rt_MatchTag;
 APTR rt_EndSkip;
 UWORD rt_Flags;
 UWORD rt_Version;
 UWORD rt_Type;
 BYTE rt_Pri;
 char *rt_Name;
 char *rt_IDString;
 APTR rt_Init;
};

struct Screen {
 struct Screen *NextScreen;
 struct Window *FirstWindow;
 SHORT LeftEdge, TopEdge;
 SHORT Width, Height;
 SHORT MouseY, MouseX;
 USHORT Flags;
 UBYTE *Title;
 UBYTE *DefaultTitle;
 BYTE BarHeight, BarVBorder, BarHBorder;
 BYTE MenuVBorder, MenuHBorder;
 BYTE WBotTop, WBotLeft, WBotRight, WBotBottom;
 struct TextAttr *Font;
 struct ViewPort ViewPort;
 struct RastPort RastPort;
 struct BitMap BitMap;
 struct Layer_Info LayerInfo;
 struct Gadget *FirstGadget;
 UBYTE DetailPen, BlockPen;
 USHORT SaveColor0;
 struct Layer *BarLayer;
 UBYTE *ExtData;
 UBYTE *UserData;
};

struct Semaphore {
 struct MsgPort sm_MsgPort;
 WORD sm_Bids;
};

struct SemaphoreRequest {
 struct MinNode sr_Link;
 struct Task *sr_Waiter;
};

struct SignalSemaphore {
 struct Node ss_Link;
 SHORT ss_NestCount;
 struct MinList ss_WaitQueue;
 struct SemaphoreRequest ss_MultipleLink;
 struct Task *ss_Owner;
 SHORT ss_QueueCount;
};
```

```
struct SimpleSprite {
 UWORD *posctldata;
 UWORD height;
 UWORD x, y;
 UWORD num;
};

struct SpriteImage {
 UWORD posctldata[2];
 UWORD sprdata[2][height];
 UWORD reserved[2];
};

struct Task {
 struct Node tc_Node;
 UBYTE tc_Flags;
 UBYTE tc_State;
 BYTE tc_IDNestCnt;
 BYTE tc_TDNestCnt;
 ULONG tc_SigAlloc;
 ULONG tc_SigWait;
 ULONG tc_SigRecvd;
 ULONG tc_SigExcept;
 UWORD tc_TrapAlloc;
 UWORD tc_TrapAble;
 APTR tc_ExceptData;
 APTR tc_ExceptCode;
 APTR tc_TrapData;
 APTR tc_TrapCode;
 APTR tc_SPReg;
 APTR tc_SPLower;
 APTR tc_SPUpper;
 VOID (*tc_Switch)();
 VOID (*tc_Launch)();
 APTR tc_UserData;
};

struct TextAttr {
 STRPTR ta_Name;
 UWORD ta_YSize;
 UBYTE ta_Style;
 UBYTE ta_Flags;
};

struct TextFont {
 struct Node TextNode;
 struct Message tf_Message;
 UWORD tf_YSize;
 UBYTE tf_Style;
 UBYTE tf_Flags;
 UWORD tf_XSize;
 UWORD tf_Baseline;
 UWORD tf_BoldSmear;
 UWORD tf_Accessors;
 UBYTE tf_LoChar;
 UBYTE tf_HiChar;
 APTR tf_CharData;
 UWORD tf_Modulo;
};
```

```
 APTR tf_CharLoc;
 APTR tf_CharSpace;
 APTR tf_CharKern;
};

struct ViewPort {
 struct ViewPort *Next;
 struct ColorMap *ColorMap;
 struct CopList *DspIns;
 struct CopList *SprIns;
 struct CopList *ClrIns;
 struct UCopList *UCopIns;
 SHORT DWidth, DHeight;
 SHORT DxOffset, DyOffset;
 UWORD Modes;
 UWORD reserved;
 struct RasInfo *RasInfo;
};

struct VSprite {
 struct VSprite *NextVSprite;
 struct VSprite *PrevVSprite;
 struct VSprite *DrawPath;
 struct VSprite *ClearPath;
 WORD OldY, OldX;
 WORD Flags;
 WORD Y, X;
 WORD Height;
 WORD Width;
 WORD Depth;
 WORD MeMask;
 WORD HitMask;
 WORD *ImageData;
 WORD *BorderLine;
 WORD *CollMask;
 WORD *SprColors;
 struct Bob *VBob;
 BYTE PlanePick;
 BYTE PlaneOnOff;
 VUserStuff VUserExt;
};

struct WBOject {
 struct Node wo_MasterNode;
 struct Node wo_Siblings;
 struct Node wo_SelectNode;
 struct Node wo_UtilityNode;
 struct WBOject *wo_Parent;
#ifdef SMARTCOMPILER
 UBYTE wo_IconDisp:1;
 UBYTE wo_DrawerOpen:1;
 UBYTE wo_Selected:1;
 UBYTE wo_Background:1;
#else;
 UBYTE woFlags;
#endif
 UBYTE wo_Type;
 USHORT wo_UseCount;
};
```

```
char *wo_Name;
SHORT wo_NameXOffset;
SHORT wo_NameYOffset;
char *wo_DefaultTool;
struct DrawerData *wo_DrawerData;
struct Window *wo_IconWin;
LONG wo_CurrentX;
LONG wo_CurrentY;
char **wo_ToolTypes;
struct Gadget wo_Gadget;
struct FreeList wo_FreeList;
char *wo_ToolWindow;
LONG wo_StackSize;
LONG wo_Lock;
};

struct Window {
 struct Window *NextWindow;
 SHORT LeftEdge, TopEdge;
 SHORT Width, Height;
 SHORT MouseY, MouseX;
 SHORT MinWidth, MinHeight;
 SHORT MaxWidth, MaxHeight;
 ULONG Flags;
 struct Menu *MenuStrip;
 UBYTE *Title;
 struct Requester *FirstRequester;
 struct Requester *DMRequest;
 SHORT ReqCount;
 struct Screen *WScreen;
 struct RastPort RPort;
 BYTE BorderLeft, BorderTop, BorderRight, BorderBottom;
 struct RastPort *BorderRPort;
 struct Gadget *FirstGadget;
 struct Window *Parent, *Descendent;
 USHORT *Pointer;
 BYTE PtrHeight;
 BYTE PtrWidth;
 BYTE XOffset, YOffset;
 ULONG IDCMPFlags;
 struct MsgPort *UserPort, *WindowPort;
 struct IntuiMessage *MessageKey;
 UBYTE DetailPen, BlockPen;
 struct Image *Checkmark;
 UBYTE *ScreenTitle;
 SHORT GZZMouseX;
 SHORT GZZMouseY;
 SHORT GZZWidth;
 SHORT GZZHeight;
 UBYTE *ExtData;
 BYTE *UserData;
};
```



# Anhang C

## System Alerts

In diesem Anhang wollen wir die System-Alerts aufführen, die mit dem Exec-Befehl »Alert(AlertNr,Para);« aufgerufen werden.

Zuerst die allgemeinen Alerts.

Für solch ein Alert muß man für AlertNr verschiedene Werte addieren:

```
Das Alert soll den Amiga neu booten: AT_DeadEnd = 0x80000000
Es ist ein IO-Fehler: AG_IOError = 0x00060000
Der Fehler ist am Audioport aufgetreten: AO_AudioDev = 0x00008010
```

Nun könnte man die verschiedenen Hexadezimalzahlen addieren, was die Zahl ergeben würde, die beim Alert mit ausgegeben wird, doch geht es viel einfacher:

```
Alert(AT_DeadEnd+AG_IOError+AOAudioDev,0);
```

Auf dem Bildschirm ist dann im Alert die Nummer zu sehen, die sich aus dieser Addition ergibt. Der Programmierer kann auf diese Weise bei jedem auftretenden Alert zurückverfolgen, an welcher Stelle der Fehler aufgetreten ist. Im Normalfall wird von jedem Typ, also AT, AG und AO, ein Wert genommen, um ihn mit den anderen zu addieren.

Anhand der folgenden Liste kann jeder auftretende Alerttyp analysiert werden:

|             |             |            |
|-------------|-------------|------------|
| Alert-Typen | AT_DeadEnd  | 0x80000000 |
|             | AT_Recovery | 0x00000000 |
| Fehler-Art  | AG_NoMemory | 0x00010000 |
|             | AG_MakeLib  | 0x00020000 |
|             | AG_OpenLib  | 0x00030000 |
|             | AG_OpenDev  | 0x00040000 |
|             | AG_OpenRes  | 0x00050000 |
|             | AG_IOError  | 0x00060000 |

|                |                 |            |
|----------------|-----------------|------------|
| Fehler-Bereich | AO_ExecLib      | 0x00008001 |
|                | AO_GraphicsLib  | 0x00008002 |
|                | AO_LayersLib    | 0x00008003 |
|                | AO_Intuition    | 0x00008004 |
|                | AO_MathLib      | 0x00008005 |
|                | AO_CListLib     | 0x00008006 |
|                | AO_DOSLib       | 0x00008007 |
|                | AO_RAMLib       | 0x00008008 |
|                | AO_IconLib      | 0x00008009 |
|                | AO_AudioDev     | 0x00008010 |
|                | AO_ConsoleDev   | 0x00008011 |
|                | AO_GamePortDev  | 0x00008012 |
|                | AO_KeyboardDev  | 0x00008013 |
|                | AO_TrackDiskDev | 0x00008014 |
|                | AO_TimerDev     | 0x00008015 |
|                | AO_CIARsrc      | 0x00008020 |
|                | AO_DiskRsrc     | 0x00008021 |
|                | AO_MiscRsrc     | 0x00008022 |
|                | AO_BootStrap    | 0x00008030 |
|                | AO_Workbench    | 0x00008031 |

Es stehen allerdings auch fertige Alerts zur Verfügung, die nur noch übernommen werden müssen, ohne addiert zu werden:

Beispielsweise `Alert(AN_ExecLib,0);` |

Spezifische Alerts:

|                  |                 |            |
|------------------|-----------------|------------|
| exec.library     | AN_ExecLib      | 0x01000000 |
|                  | AN_ExcptVect    | 0x81000001 |
|                  | AN_BaseChkSum   | 0x81000002 |
|                  | AN_LibChkSum    | 0x81000003 |
|                  | AN_LibMem       | 0x81000004 |
|                  | AN_MemCorrupt   | 0x81000005 |
|                  | AN_IntrMem      | 0x81000006 |
|                  | AN_InitAPtr     | 0x81000007 |
| graphics.library | AN_GraphicsLib  | 0x02000000 |
|                  | AN_CopDisplay   | 0x82000001 |
|                  | AN_CopInstr     | 0x82000002 |
|                  | AN_CopListOver  | 0x82000003 |
|                  | AN_CopIListOver | 0x82000004 |
|                  | AN_CopListHead  | 0x82000005 |
|                  | AN_LongFrame    | 0x82000006 |
|                  | AN_ShortFrame   | 0x82000007 |
|                  | AN_FloodFill    | 0x82000008 |
|                  | AN_TextTmpRas   | 0x02000009 |
|                  | AN_BitBitMap    | 0x8200000A |
| layers.library   | AN_LayersLib    | 0x03000000 |

|                   |                 |            |
|-------------------|-----------------|------------|
| intuition.library | AN_Intuition    | 0x04000000 |
|                   | AN_GadgetType   | 0x84000001 |
|                   | AN_BadGadget    | 0x04000001 |
|                   | AN_CreatePort   | 0x84000002 |
|                   | AN_ItemAlloc    | 0x04000003 |
|                   | AN_SubAlloc     | 0x04000004 |
|                   | AN_PlaneAlloc   | 0x84000005 |
|                   | AN_ItemBoxTop   | 0x84000006 |
|                   | AN_OpenScreen   | 0x84000007 |
|                   | AN_OpenScrnRast | 0x84000008 |
|                   | AN_SysScrnType  | 0x84000009 |
|                   | AN_AddSWGadget  | 0x8400000A |
|                   | AN_OpenWindow   | 0x8400000B |
|                   | AN_BadState     | 0x8400000C |
|                   | AN_BadMessage   | 0x8400000D |
|                   | AN_WeirdEcho    | 0x8400000E |
| AN_NoConsole      | 0x8400000F      |            |
| math.library      | AN_MathLib      | 0x05000000 |
| clist.library     | AN_CListLib     | 0x06000000 |
| dos.library       | AN_DOSLib       | 0x07000000 |
|                   | AN_StartMem     | 0x07000001 |
|                   | AN_EndTask      | 0x07000002 |
|                   | AN_QPktFail     | 0x07000003 |
|                   | AN_AsyncPkt     | 0x07000004 |
|                   | AN_FreeVec      | 0x07000005 |
|                   | AN_DiskBlkSeq   | 0x07000006 |
|                   | AN_BitMap       | 0x07000007 |
|                   | AN_KeyFree      | 0x07000008 |
|                   | AN_BadChkSum    | 0x07000009 |
|                   | AN_DiskError    | 0x0700000A |
|                   | AN_KeyRange     | 0x0700000B |
| AN_BadOverlay     | 0x0700000C      |            |
| ramlib.library    | AN_RAMLib       | 0x08000000 |
| icon.library      | AN_IconLib      | 0x09000000 |
| audio.device      | AN_AudioDev     | 0x10000000 |
| console.device    | AN_ConsoleDev   | 0x11000000 |
| gameport.device   | AN_GamePortDev  | 0x12000000 |
| keyboard.device   | AN_KeyboardDev  | 0x13000000 |
| trackdisk.device  | AN_TrackDiskDev | 0x14000000 |
|                   | AN_TDCalibSeek  | 0x14000001 |
|                   | AN_TDDelay      | 0x14000002 |
| timer.device      | AN_TimerDev     | 0x15000000 |
|                   | AN_TMBadReq     | 0x15000001 |
| cia.resource      | AN_CIARsrc      | 0x20000000 |

|               |               |            |
|---------------|---------------|------------|
| disk.resource | AN_DiskRsrc   | 0x21000000 |
|               | AN_DRHasDisk  | 0x21000001 |
|               | AN_DRIntNoAct | 0x21000002 |
| misc.resource | AN_MiscRsrc   | 0x22000000 |
| bootstrap     | AN_BootStrap  | 0x30000000 |
|               | AN_BootError  | 0x30000001 |
| workbench     | AN_Workbench  | 0x31000000 |

# Anhang D

## Die DOS-Fehlermeldungen

Mit dem Befehl IoErr kann der zuletzt aufgetretene Ein-/Ausgabe-Fehler abgefragt werden. IoErr gibt dabei eine Fehlernummer zurück. Diese Fehlernummer kann anhand der folgenden Liste in Klartext umgewandelt werden:

| Fehlernummer | Fehlertext englisch/deutsch                                                        |
|--------------|------------------------------------------------------------------------------------|
| 103          | insufficient free store<br>Nicht genügend freier Speicherplatz                     |
| 104          | task table full<br>Zuviele Tasks: Es können maximal 20 CLI-Tasks laufen            |
| 120          | argument line invalid or too long<br>Die Befehlszeile ist falsch oder ist zu lang  |
| 121          | file is not an object module<br>Die Datei ist nicht ladbar                         |
| 122          | invalid resident library during load<br>Fehler in einer Befehls-Bibliothek         |
| 202          | object in use<br>Eine Datei kann nicht beschrieben und gleichzeitig gelesen werden |
| 203          | object already exists<br>Datei besteht schon                                       |
| 204          | directory not found<br>Verzeichnis nicht gefunden                                  |
| 205          | object not found<br>Datei nicht gefunden                                           |
| 206          | invalid window<br>Ungültige Window-Dimension oder -Zuweisung                       |

|     |                                                                                |
|-----|--------------------------------------------------------------------------------|
| 209 | packet request type unknown<br>Ungültiger Device-Befehl                        |
| 210 | invalid stream component name<br>Dateiname zu lang oder mit ungültigen Zeichen |
| 211 | invalid object lock<br>Keine gültige Lock-Structure                            |
| 212 | object not of required type<br>Ungültiger Typ (type dir ist nicht zulässig)    |
| 213 | disk not validated<br>Disk-Fehler oder Disk-Erkennung nicht abgeschlossen      |
| 214 | disk write-protected<br>Diskette ist schreibgeschützt                          |
| 215 | rename across devices attempted<br>Dateinamenänderung nicht gültig             |
| 216 | directory not empty<br>Verzeichnis nicht leer                                  |
| 218 | device not mounted<br>Log. Gerät nicht auffindbar                              |
| 219 | seek error<br>Seek-Befehl mit ungültigen Parametern                            |
| 220 | comment too big<br>Kommentar zu lang                                           |
| 221 | disk full<br>Diskette ist voll                                                 |
| 222 | file is protected from deletion<br>Datei ist vor löschen geschützt             |
| 223 | file is protected from writing<br>Datei ist vor überschreiben geschützt        |
| 224 | file is protected from reading<br>Datei ist vor lesen geschützt                |
| 225 | not a DOS disk<br>Diskette nicht mit DOS-Format                                |
| 226 | no disk in drive<br>Keine Diskette im Laufwerk                                 |

232

no more entries in directory  
Im Verzeichnis sind keine weiteren Dateien mehr  
vorhanden



# Anhang E

## Anmerkungen zur Programmgestaltung

Commodore hat einige Rahmenrichtlinien erstellt, an die sich jeder Amiga-Programmierer halten sollte. Diese wollen wir an dieser Stelle aufführen:

- MENU** Menüpunkte, die im Programm momentan keine Bedeutung haben, sollten mit dem Befehl »OffMenu()« abgeschaltet werden, da der Benutzer einen solchen Menüpunkt nicht anwählen soll, wenn anschließend nichts geschieht.
- MENU** Benötigt Ihr Programm mehrere Windows, mit unterschiedlichen Menüs, sollten diese farblich unterschieden werden.
- MENU** Arbeitet Ihr Programm mit Dateien, sollte das »LOAD/SAVE«-Menü folgende Menüpunkte besitzen:
- |                  |                                                               |
|------------------|---------------------------------------------------------------|
| <b>NEW</b>       | Neue Datei erstellen                                          |
| <b>OPEN</b>      | Alte Datei laden                                              |
| <b>SAVE</b>      | Datei unter dem Namen speichern, unter dem sie geladen wurde. |
| <b>SAVE AS..</b> | Datei unter neuem Namen speichern                             |
| <b>PRINT</b>     | Datei drucken                                                 |
| <b>PRINT AS</b>  | Teil einer Datei drucken oder neue Druckerparameter wählen    |
| <b>QUIT</b>      | Beenden des Programms                                         |
- GADGET** Wenn über Gadgets abgefragt wird, ob das Programm eine Handlung durchführen soll, sollte das Gadget, das dieses beneint, immer hervorgehoben werden.
- GADGET** Gadgets sollten sich niemals überlappen.

|           |                                                                                                                                                                                                                                                                                                         |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GADGET    | Genauso wie Menüs, sollten auch Gadgets mit OffGadget abgeschaltet werden, wenn sie keine Funktion mehr haben.                                                                                                                                                                                          |
| REQUESTER | Es sollte immer eine Möglichkeit gegeben werden, diesen Requester zu verlassen. Zum Beispiel sollte bei einem Requester, der abfragt, ob die eingelegte Diskette wirklich formatiert werden soll, auf jeden Fall ein Gadget vorhanden sein, das dies verneint. Dieses Gadget sollte hervorgehoben sein. |
| REQUESTER | Bei Requestern, die nur zwei Auswahlmöglichkeiten bieten, sollte immer die sichere Option auf der rechten Seite erscheinen, während auf der linken Seite die Durchführung der entsprechenden Funktion vom Benutzer bejaht wird.                                                                         |
| MAUS      | Für Abfragen sollte immer die linke Maustaste verwendet werden, während die rechte für Menüabfragen zur Verfügung steht.                                                                                                                                                                                |

# Anhang F

## Drucker-Codes

Mittels der Printer-Device kann der Drucker eingestellt werden. Dazu muß allerdings in der Structure angegeben werden, welche Einstellung geändert werden soll und eventuell müssen auch noch die neuen Parameter angegeben werden.

Steht bei den Escape-Funktionen ein »n«, so müssen Parameter mit angegeben werden:

| CODE    | NR. | ESCAPE  | BEMERKUNG                  |
|---------|-----|---------|----------------------------|
| aRIs    | 0   | ESCc    | Rücksetzung des Druckers   |
| aRIN    | 1   | ESC#1   | Initialisierung            |
| aIND    | 2   | ESCD    | Zeilenvorschub             |
| aNEL    | 3   | ESCE    | Return und Zeilenvorschub  |
| aRI     | 4   | ESCM    | Umgekehrter Zeilenvorschub |
| aSGR0   | 5   | ESC[0m  | Normaler Zeichensatz       |
| aSGR3   | 6   | ESC[3m  | Kursivschrift ein          |
| aSGR23  | 7   | ESC[23m | Kursivschrift aus          |
| aSGR4   | 8   | ESC[4m  | Unterstreichen ein         |
| aSGR24  | 9   | ESC[24m | Unterstreichen aus         |
| aSGR1   | 10  | ESC[1m  | Fettdruck ein              |
| aSGR22  | 11  | ESC[22m | Fettdruck aus              |
| aSFC    | 12  | ESC[nm  | Vordergrundfarbe setzen    |
| aSBC    | 13  | ESC[nm  | Hintergrundfarbe setzen    |
| aSHORP0 | 14  | ESC[0w  | Normaler Zeichenabstand    |
| aSHORP2 | 15  | ESC[2w  | Elite ein                  |
| aSHORP1 | 16  | ESC[1w  | Elite aus                  |
| aSHOPR4 | 17  | ESC[4w  | Kleinschrift ein           |
| aSHORP3 | 18  | ESC[3w  | Kleinschrift aus           |
| aSHORP6 | 19  | ESC[6w  | Breitschrift ein           |
| aSHORP5 | 20  | ESC[5w  | Breitschrift aus           |
| aDEN6   | 21  | ESC[6"z | Schattendruck ein          |
| aDEN5   | 22  | ESC[5"z | Schattendruck aus          |
| aDEN4   | 23  | ESC[4"z | Doppeldruck ein            |
| aDEN3   | 24  | ESC[3"z | Doppeldruck aus            |
| aDEN2   | 25  | ESC[2"z | Schönschrift ein           |
| aDEN1   | 26  | ESC[1"z | Schönschrift aus           |
| aSUS2   | 27  | ESC[2v  | Hochstellung ein           |
| aSUS1   | 28  | ESC[1v  | Hochstellung aus           |
| aSUS4   | 29  | ESC[4v  | Tiefstellung ein           |
| aSUS3   | 30  | ESC[3v  | Tiefstellung aus           |
| aSUS0   | 31  | ESC[0v  | Zeilen-Normalisierung      |
| aPLU    | 32  | ESCL    | Zeile auf                  |
| aPLD    | 33  | ESCK    | Zeile ab                   |

|         |    |            |                                     |
|---------|----|------------|-------------------------------------|
| aFNT0   | 34 | ESC(B      | US - Zeichensatz                    |
| aFNT1   | 35 | ESC(R      | Französischer Zeichensatz           |
| aFNT2   | 36 | ESC(K      | Deutscher Zeichensatz               |
| aFNT3   | 37 | ESC(A      | Englischer Zeichensatz              |
| aFNT4   | 38 | ESC(E      | Dänisch1 Zeichensatz                |
| aFNT5   | 39 | ESC(H      | Schwedischer Zeichensatz            |
| aFNT6   | 40 | ESC(Y      | Italienischer Zeichensatz           |
| aFNT7   | 41 | ESC(Z      | Spanischer Zeichensatz              |
| aFNT8   | 42 | ESC(J      | Japanischer Zeichensatz             |
| aFNT9   | 43 | ESC(6      | Norwegischer Zeichensatz            |
| aFNT10  | 44 | ESC(C      | Dänisch2 Zeichensatz                |
| aPROP2  | 45 | ESC[2p     | Proportionalschrift ein             |
| aPROP1  | 46 | ESC[1p     | Proportionalschrift aus             |
| aPROP0  | 47 | ESC[0p     | Proportionalschrift löschen         |
| aTSS    | 48 | ESC[n E    | Proportionalschrift-Offset setzen   |
| aJFY5   | 49 | ESC[5 F    | Linksbündiger Druck                 |
| aJFY7   | 50 | ESC[7 F    | Rechtsbündiger Druck                |
| aJFY6   | 51 | ESC[6 F    | Blocksatz ein                       |
| aJFY0   | 52 | ESC[0 F    | Zeilen-Justierung aus               |
| aJFY3   | 53 | ESC[3 F    | Zeichenbreite setzen                |
| aJFY1   | 54 | ESC[1 F    | Zeilenzentrierung                   |
| aVERP0  | 55 | ESC[0z 1/8 | Zeilenschaltung                     |
| aVERP1  | 56 | ESC[1z 1/6 | Zeilenschaltung                     |
| aSLPP   | 57 | ESC[nt     | Papierlänge setzen                  |
| aPERF   | 58 | ESC[nq     | Blattendeübersprung ein             |
| aPERF0  | 59 | ESC[0q     | Blattendeübersprung aus             |
| aLMS    | 60 | ESC#9      | Linken Rand setzen                  |
| aRMS    | 61 | ESC#0      | Rechten Rand setzen                 |
| aTMS    | 62 | ESC#8      | Oberen Rand setzen                  |
| aBMS    | 63 | ESC#2      | Unteren Rand setzen                 |
| aSTBM   | 64 | ESC[n;nr   | Oberen und unteren Rand setzen      |
| aSLRM   | 65 | ESC[n;ns   | Linken und rechten Rand setzen      |
| aCAM    | 66 | ESC#3      | Ränder löschen                      |
| aHTS    | 67 | ESCH       | Horizontale Tabs setzen             |
| aVTS    | 68 | ESCJ       | Vertikale Tabs setzen               |
| aTBC0   | 69 | ESC[0g     | Horizontalen Tab löschen            |
| aTBC3   | 70 | ESC[3g     | Alle horizontalen Tabs löschen      |
| aTBC1   | 71 | ESC[1g     | Vertikalen Tab löschen              |
| aTBC4   | 72 | ESC[4g     | Alle Vertikalen Tabs löschen        |
| aTBCALL | 73 | ESC#4      | Alle horiz. und vert. Tabs löschen. |
| aTBSALL | 74 | ESC#5      | Normale Tabs setzen                 |
| aEXTEND | 75 | ESC[n"x    | Erweiterungsbefehle                 |

# Anhang G

## Die Demo-Diskette

Auf der beiliegenden Diskette befinden sich eine Vielzahl von Demonstrationen zu den verschiedenen Themen in diesem Buch. Diese Programme sind in fünf Verzeichnisse eingeteilt. Diese stimmen mit denen aus diesem Buch (siehe Inhaltsverzeichnis) überein. Wenn Sie auf diese Programme zugreifen wollen, so müssen Sie als Erstes in das CLI. Wie dies geschieht, ist ausführlich in Kapitel 1 beschrieben. Anschließend können Sie sich das Inhaltsverzeichnis mittels »dir "m&t demodisk" opt a« ausgeben lassen. Sie sehen nun alle Einträge auf der Diskette.

Zusätzlich zu den fünf Inhaltsverzeichnissen befinden sich noch weitere Dateien auf der Diskette. Diese haben folgende Bedeutung:

|            |                                                                                                                                                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Converter  | Dies ist das Konvertierungsprogramm aus Kapitel 10. Es läßt sich nur starten, wenn gleichzeitig Basic aufgerufen werden kann. Zum Benutzen sollten Sie es also auf Ihre Basic-Diskette kopieren (»copy "m&t demodisk:converter" to xxxx«). |
| IFFBild    | Diese Datei enthält das Bild, das mit der IFF-Read-Demonstration eingeladen wird. Dazu muß sich die Diskette allerdings in dem Laufwerk »df0:« befinden.                                                                                   |
| testbild   | Kompiliertes C-Programm, zum Einstellen des Monitors.                                                                                                                                                                                      |
| testbild.c | Source-Code von »testbild«.                                                                                                                                                                                                                |
| Titelbild  | Bild-File, das durch die Dosedemo eingelesen wird. Dazu muß sich allerdings die Diskette in Laufwerk »df0:« befinden.                                                                                                                      |

Zudem befinden sich noch einige .info-Dateien auf der Diskette, die die Icons für die »Scublade« enthalten.

Die eigentlichen C-Programme befinden sich in den Verzeichnissen, wenn man von »testbild« einmal absieht. Auf diese Programme kann recht einfach zugegriffen werden. Am vorteilhaftesten ist es, wenn Sie als Erstes die Demonstrationsdiskette als aktuelles Verzeichniss setzen. Dies geschieht mit »cd "m&t demodisk"«. Wenn Sie nun die Demonstration »hamdemo« aktivieren wollen, so müssen Sie wissen, in welchem Verzeichniss es sich befindet. Mit »dir opt a« geht dies leicht. Aufrufen können Sie die Demonstration nun durch »grafikgrund/handemo«. In der Datei »handemo.c« befindet sich der Source-Code zum Programm. Ihn können Sie mit »ed grafikgrund/handemo.c« editieren. Wollen Sie anschließend den Editor verlassen, ohne das Programm wieder zu speichern, so drücken Sie die »ESC«-Taste, »q« und anschließend »RETURN«.

Wollen Sie ein Programm nicht nur vom CLI, sondern auch von der Workbench aus starten, so müssen Sie auf dieser das Icon »M&T DemoDisk« anklicken. Anschließend erscheinen fünf »Schubladen«, in denen sich die Programme befinden. Allerdings besitzen nicht alle Programme ein Icon, da diese dann nicht von der Workbench aus zu starten sind.

Folgende Programme befinden sich auf Ihrer Demonstrationsdiskette (In Klammern stehen die Dateien mit dem gleichen Namen, aber mit Kürzel .c oder .info angehängt. Ist das Kürzel .info vorhanden, so kann das Programm auch von der Workbench aus gestartet werden.):

#### **Verzeichnis SONDERTEIL:**

|                                |                                                                                  |
|--------------------------------|----------------------------------------------------------------------------------|
| iffreaddemo<br>(.c und .info)  | Liest das Bild IFFBILD im IFF-Format ein.                                        |
| iffwritedemo<br>(.cund.info)   | Öffnet einen Screen und schreibt diesen unter dem Namen Demoscreen auf Diskette. |
| mathdemo<br>(c. und .info)     | Führt eine Anzahl Berechnungen mittels der Math-Libraries durch.                 |
| multidemo<br>(.cund.info)      | Öffnet zwei Windows, in die unabhängig voneinander gezeichnet wird.              |
| narratordemo<br>(.c und .info) | Spricht einen Satz zu Ihnen.                                                     |

**Verzeichnis GRAFIKTEIL:**

|                               |                                                                         |
|-------------------------------|-------------------------------------------------------------------------|
| areademo<br>(.c und .info)    | Zeichnet zwei Polygone auf einen Screen.                                |
| graphicdemo<br>(.c und .info) | Zeigt die Möglichkeiten verschiedener Grafikbefehle.                    |
| imagedemo<br>(.c und .info)   | Demonstration für Images und Borders.                                   |
| pointerdemo<br>(.c und .info) | Zeigt eine Animationsmöglichkeit mit dem Mauszeiger.                    |
| prefdemo<br>(.c und .info)    | Verschiebt sämtliche Screens hardwaremäßig.                             |
| spritedemo<br>(.c und .info)  | Animation mit Hardware-Sprites.                                         |
| textdemo<br>(.c und .info)    | Zeigt die Möglichkeiten der Verwendung von verschiedenen Zeichensätzen. |
| vspritedemo<br>(.c und .info) | Zeigt die Animation mit VSprites.                                       |

**Verzeichnis GRAFIKGRUND:**

|                                |                                             |
|--------------------------------|---------------------------------------------|
| halfbriedemo<br>(.c und .info) | 64 Farben auf dem Amiga.                    |
| hamdemo<br>(.c und .info)      | Die Steigerung: 4096 Farben gleichzeitig.   |
| screendemo<br>(.c und .info)   | Zeigt die Verwendung von Screens.           |
| windowdemo<br>(.c und .info)   | Verschiedene Windowtypen auf der Workbench. |

**Verzeichnis EINAUSGABE:**

|                               |                                                                                                    |
|-------------------------------|----------------------------------------------------------------------------------------------------|
| dosdemo<br>(nur .c)           | Lädt ein Bild von Diskette. Die Demonstrationsdiskette muß sich dabei im Laufwerk »df0:« befinden. |
| dosprinterdemo<br>(nur .c)    | Gibt einen kurzen Text auf dem Drucker aus.                                                        |
| druckerdemo<br>(.c und .info) | Erweiterte Druckerkontrolle über die Device.                                                       |

dumpdemo  
(.c und .info)

einfache Hardcopy-Routine.

**Verzeichnis BEDIENUNG:**

alertsdem  
(.c und .info)

Zeigt die Verwendung von Intuition-Alerts.

autoreqdemo  
(.c und .info)

Einfacher Requester.

dmreqdemo  
(.cund.info)

Double-Menü-Requester. Dazu muß nach dem Starten des Programms zuerst zweimal die rechte Maustaste betätigt werden, damit der Requester erscheint.

excalertdemo  
(.c und .info)

Zeigt zwei Exec-Alerts.

gadgetdemo  
(.cund.info)

Die verschiedenen Möglichkeiten für User-Gadgets

menudemo  
(.c und .info)

Zeigt die Variationsmöglichkeiten bei Menüs.

# Stichwortverzeichnis

## A

ActivateWindow 75  
AddFont 117  
AddFreeList 288  
AddGadget 218  
AddVSprite 154  
Alerts 182, 231f  
AllocWBOBJECT 289  
AMIGA-DOS 277  
Animation 139, 168, 174  
AnimObjects 139  
Arbeitsoberfläche 287  
Area 94  
AreaCircle 94  
AreaDraw 95  
AreaEllipse 95  
AreaEnd 96  
AreaMove 97  
AskFont 118  
AskSoftStyle 119  
Ausdruck 279  
AUTOKNOB 215  
AutoRequest 244  
AvailFonts 119

## B

Backdrop-Window 68  
Bedingungen 24  
BEEPING 50  
BeginRefresh 76  
Benutzeroberfläche 181  
Betriebssystem 181, 305  
Bibliotheken 29  
Bildfrequenz 39  
Bildschirm 39, 174  
BitMap 51, 67  
BitMapHeader 332  
BitPlanes 40, 130  
Blitter-Objekts 139  
BMHD 332  
BNDRYOFF 97

Bob 139  
BODY 332  
Boolean-Gadget 201, 210  
Borderless-Window 66  
Borders 93, 132  
BuildSysRequest 245  
BumpRevision 290

## C

CAMG 332  
CCRT 332  
ChangePri 307  
ChangeSprite 143  
Checkmark 182  
CHIP-Memory 39  
ClearDMRequest 246  
ClearEOL 121  
ClearMenuStrip 188  
ClearPointer 76, 170  
ClearScreen 121  
CLI 181, 277, 306  
clist.library 345  
Close 257  
Close-Gadget 71  
CloseFont 121  
CloseScreen 52  
CloseWindow 77  
CloseWorkbench 52  
CMAP 332  
CMHD 332  
ColorMap 332  
ColorRange 332  
Command-Tasten 193  
Copper-Liste 40  
CPU 39  
CreateDir 257  
CreateTask 308  
CRNG 332  
CurrentDir 258  
CurrentTime 342  
Custom-Chips 39

Custom-Screens 41, 45  
CUSTOMBITMAP 50  
CUSTOMSCREEN 50  
CycleInfo 332

## D

Datentyp 22  
DeadEnd-Alerts 232  
DeleteFile 258  
DeleteTask 309  
Devices 34  
Diskette 271, 287  
diskfont.library 346  
DisplayBeep 53  
doppelte Genauigkeit 313f  
DOS 255, 371  
dos.library 346  
DoubleClick 343  
DPPV 332  
Draw 98, 117  
DrawCircle 98  
DrawEllipse 99  
DrawGList 155  
DrawImage 129  
Drucker 277  
DUALPF 42  
DupLock 259

## E

EndRefresh 78  
EndRequest 246  
Examine 259  
Exec 232, 367  
exec.library 347  
Execute 260  
exec\_support.library 348  
ExNext 261  
Extra-Halfbright 40  
EXTRA\_HALFBRIGHT 44, 46

## F

Farben 40  
Farbregler 40  
Farbtabelle 40  
FAST-Memory 39  
Fenster 39  
FFP-Zahlen 313  
Flags 50  
Flood 99  
FORM 332  
FreeDiskObject 291  
FreeFreeList 292

FreeSprite 144  
FreeSysRequest 246  
FreeWBOject 293

## G

Gadget 65, 71, 93, 129, 132, 182, 201, 375  
Gadget-Typ 182  
GENLOCK\_VIDEO 44  
GetDefPrefs 175  
GetDiskObject 294  
GetIcon 295  
GetPrefs 176  
GetRGB4 100  
GetSprite 145  
GetWBOject 296  
gfxmacros 354  
Gimmezerozero-Window 66  
Gleitkomma-Zahlen 313  
Grafiken 193  
graphics.library 348

## H

HAM 40, 42f, 46  
Hardcopy 279  
Hardware-Sprites 139  
HIRES 42, 43, 46  
Hold-And-Modify 40, 42  
Hot-Spot 168

## I

Icon 287  
IDCMP 71, 194, 216  
IFF 331  
ILBM 332  
Images 93, 129  
Info 261  
InitArea 100  
InitGels 156  
InitMasks 157  
InitRequest 247  
InitTmpRas 101  
Input 262  
Interchange-File-Format 331  
Interlace 39, 42  
IntuiText 137  
Intuition 39, 71  
Intuition-Alert 235  
Intuition-Text 137  
intuition.library 351  
IOError 262, 371  
IsInteractive 262  
ItemAddress 189

ITEMNUM 189  
 Items 186  
 ITEMTEXT 193

## K

Kickstart 175  
 Kollisionsabfrage 153

## L

LACE 43, 46  
 Lattice-C 15  
 layers.library 352  
 LoadView 157  
 Lock 263

## M

Macros 354  
 MakeScreen 53  
 mathffp-Library 314  
 mathffp.library 353  
 mathieedoubbas.library 314, 354  
 mathtrans.library 353  
 Maus 140, 168, 182  
 MAUS 376  
 Mauszeiger 141, 185  
 MENU 375  
 MenuItem 186  
 Menükasten 186  
 MENUNUM 190  
 Menüoberbegriff 182, 186  
 Menüpunkt 185  
 Menüs 182, 185  
 Menüschalter 182  
 Menüunterpunkte 182  
 Menüzeile 186  
 ModifyIDCMP 79  
 ModifyProp 218  
 Move 102, 117  
 MoveScreen 54  
 MoveSprite 145  
 MoveWindow 80  
 MrgCop 157  
 Multitasking 65, 305  
 MutualExclude 193

## N

Narrator-Device 301  
 NewScreen 41, 45  
 NewWindow 72  
 NOCAREREFRESH 69  
 Normal-Window 66

## O

OffGadget 220  
 OffMenu 190  
 OFF\_DISPLAY 102, 354  
 OFF\_SPRITE 146, 354  
 OnGadget 221  
 OnMenu 191  
 ON\_DISPLAY 103, 354  
 ON\_SPRITE 146, 354  
 Open 264  
 OpenDiskFont 122  
 OpenFont 123  
 OpenScreen 47, 54  
 OpenWindow 80  
 OpenWorkBench 55  
 Output 264

## P

ParentDir 265  
 PlaneOnOff 130  
 PlanePick 130  
 Playfields 42, 140  
 PolyDraw 103  
 Polygone 93  
 Preferences 140, 174, 277  
 PrintText 137  
 Programmbedienung 181  
 Proportional-Gadget 201, 214  
 Pull-Down-Menü 65, 71  
 PutDiskObject 297  
 PutIcon 298  
 PutWBOject 299

## R

RastPort 40, 49, 51, 133  
 Read 265  
 ReadPixel 104  
 Rechtecke 93  
 Recovery-Alerts 232  
 RectFill 104  
 RefreshGadgets 221  
 RefreshWindowFrame 81  
 RemakeDisplay 55  
 RemFont 123  
 RemoveGadget 222  
 RemTask 310  
 RemVSprite 158  
 Rename 266  
 ReportMouse 82  
 Request 247  
 Requester 183, 237, 376  
 RethinkDisplay 56

**S**

Scheduling 305  
Schleifen 26  
Schriften 117  
Screen 39, 47, 49, 93  
ScreenToBack 56  
ScreenToFront 56  
ScrollRaster 105  
ScrollVPort 105  
Seek 267  
SetAfPt 106  
SetAPen 107  
SetBPen 107  
SetComment 268  
SetDMRequest 248  
SetDrMd 108  
SetDrPt 109  
SetFont 123  
SetMenuStrip 191  
SetOPen 109  
SetPointer 82, 140, 168  
SetPrefs 176  
SetProtection 268  
SetRast 109  
SetRGB4 48, 57, 110  
SetSoftStyle 124  
SetWindowTitles 84  
SetWrMsk 110  
SHOWTITLE 50  
ShowTitle 58  
SIMPLEREFRESH 69  
SizeWindow 85  
SMARTREFRESH 69  
SortGList 158  
SPAbs 317  
SPAcos 321  
SPAdd 317  
SPAsin 321  
SPAtan 321  
SPCmp 316  
SPCos 322  
SPCosh 324  
SPDiv 319  
Speicher 39  
SPExp 325  
SPFieee 327  
SPFix 315  
SPFlt 315  
SPLog 325  
SPLog10 326  
SPMul 318  
SPNeg 317

SPPow 326  
Sprachausgabe 301  
Sprites 42, 44, 139, 141  
SPSin 322  
SPSincos 323  
SPSinh 324  
SPSqrt 327  
SPSub 318  
SPTan 323  
SPTanh 324  
SPTieee 327  
SPTst 316  
Steuerzeichen 279  
Structures 355  
Strukturen 27  
Subitem 186, 193  
SUBNUM 192  
SUPERBITMAP 69  
SuperBitMap-Window 67  
System Alerts 367  
System-Gadgets 65, 70, 201f  
System-Meldungen 231  
System-Requester 231

**T**

Tasks 305  
Text 117, 125, 137  
TEXT-Gadgets 202  
Text/Integer-Gadget 211  
Textcursor 117  
Texte 93  
TextLength 125  
Translator-Library 301  
Transzendente Funktionen 320

**U**

Umrahmungen 132  
Unlock 269

**V**

VBeamPos 111  
View-Modi 43  
ViewPort 40, 49, 51  
ViewPortAddress 85  
virtuelle Sprites 139, 152  
VP\_HIDE 44  
VSprites 139, 152

**W**

WaitBOVP 111  
WaitForChar 269  
WaitTOF 112, 159

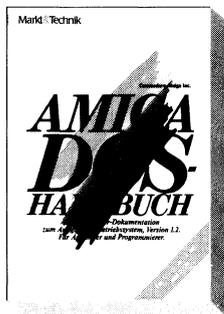
WBENCHSCREEN 50  
WBenchToBack 58  
WBenchToFront 59  
Window 39, 65, 93  
Window-Befehle 72  
Window-Gadgets 70  
Window-Refreshing 69  
Window-Typ 66  
WINDOWCLOSE 70  
WINDOWDEPTH 70  
WINDOWDRAG 70

WindowLimits 86  
WINDOWSIZE 70  
WindowToBack 87  
WindowToFront 87  
Workbench 141, 175, 287, 306  
Write 270  
WritePixel 112

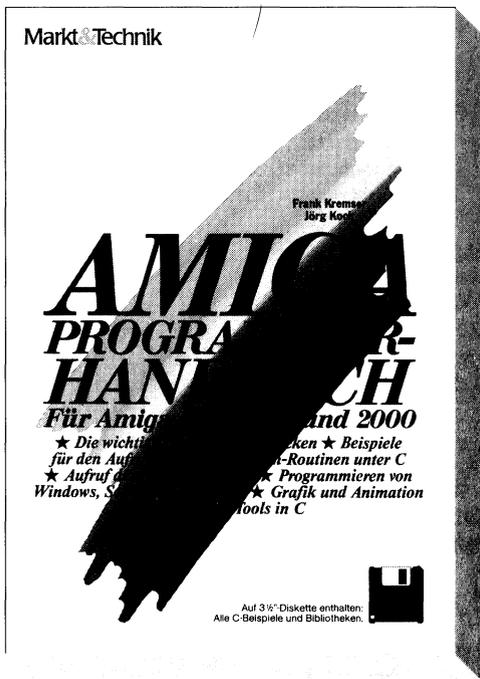
## **Z**

Zeiger 23  
Zeilensprungverfahren 39, 43

# Bücher zum Amiga

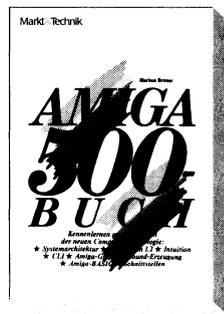


Commodore-Amiga Inc.  
**Das Amiga-DOS-Handbuch für Amiga 500, 1000 und 2000**  
 1988, 342 Seiten  
 Die Pflichtlektüre für jeden Commodore-Amiga-Anwender und Programmierer: eine Entwickler-Dokumentation zum Amiga-DOS-Betriebssystem, Version 1.2. Programmierung, interne Datenstruktur und Diskettenhandlung. Mit diesem Buch lernen Sie das mächtige Amiga DOS schnell und sicher zu beherrschen. Alle Möglichkeiten des Systems, bis hin zum »Multi-Tasking« werden ausführlich und anschaulich beschrieben.  
**Best.-Nr. 90465**  
**ISBN 3-89090-465-3**  
**DM 59,-**



Kremser/Koch  
**Amiga Programmierhandbuch**  
 1987, 387 Seiten,  
 inkl. Diskette  
 Eine tolle Einführung in die »Interna« des Amiga: Die wichtigsten Systembibliotheken, die das Betriebssystem zur Verfügung stellt, werden anhand vieler Bei-

spiele erklärt. Aus dem Inhalt: Aufruf der Betriebssystem-Routinen unter C, Aufruf der DOS-Funktionen, Programmieren von Windows, Screens und Gadgets, Grafik und Animation, Tips und Tools in C.  
**Best.-Nr. 90491**  
**ISBN 3-89090-491-2**  
**DM 69,-**



M. Breuer  
**Das Amiga-500-Handbuch**  
 1987, 489 Seiten  
 Eine ausführliche Einführung in die Bedienung des Amiga 500. Kennenlernen und Anwenden der neuen Computer-Technologie: Systemarchitektur, Workbench 1.2, Intuition, CLI, Amiga-Grafik, Sound-Erzeugung, Amiga-BASIC und Schnittstellen. Neben dem Handbucheil mit vielen Bildschirmfotos und Übersichtstabellen, die Ihnen beim täglichen Einsatz helfen, schnell und reibungslos zu arbeiten, enthält das Buch eine ausführliche Beschreibung des Amiga 500 und seines Zubehörs.  
**Best.-Nr. 90522**  
**ISBN 3-89090-522-6**  
**DM 49,-**



# Amiga-Software

# ZING!

## Das mächtige CLI-Werkzeug

Haben Sie das Eintippen satt? Zing! ermöglicht Ihnen den mausgestützten Zugriff auf Ihr Amiga-Betriebssystem.

Dieses Programm übernimmt die lästige und fehleranfällige Tipparbeit beim Arbeiten mit dem Betriebssystem Ihres Amiga. Zing! befindet sich nach dem erstmaligen Abrufen im Hintergrund und kann mit Hilfe von sogenannten »Hotkeys« jederzeit in Aktion treten. Volle Multitasking-Fähigkeit ist selbstverständlich. Wahlweise über Maus oder Funktionstasten stehen Ihnen speicherresident unter anderem folgende Funktionen zur Verfügung:

- Verzeichnis wechseln - Anzeigen eines Dateibaums - Dateien kopieren - Dateien umbenennen - Dateien schreibschützen - Restspeicheranzeige - Dateien löschen - Dateien zusammenführen - Dateien verlagern - Verzeichnisse erstellen - Dateikommentar erstellen - Systemstatusanzeige - automatische Bildschirmabschaltung (Screen Saver) ... und vieles mehr! Die Auswahl der Dateien kann mit der Maus vorgenommen werden, mögliche Kriterien sind zum Beispiel auf Dateinamen



basierende Sortiermuster oder der Zeitpunkt der Dateierstellung. Verzeichnisanzeige mit Schnellsortierdurchlauf ist bei Zing! genauso selbstverständlich wie die Möglichkeit, sowohl ganze Dateibäume als auch Teile von ihnen zu kopieren. Zusätzlich enthält das Programm viele nützliche Dienstprogramme, zum Beispiel:

- Druckerspooles - Bildschirmausdruck - Speichern eines Bildschirms als IFF-Grafik
- Überwachung von anderen Programmen
- Umbelegung der Funktionstasten - interne Symbolzuweisung
- Diskcopy-Funktion - Disketten installieren - Disketten umbenennen - Disketten formatieren - direkter Aufruf von Programmen

#### Lieferumfang:

- deutsche Programmversion auf 3 1/2"-Diskette
- Handbuch deutsch

#### Hardware-Anforderungen:

- Amiga 500, 1000 oder 2000

#### Software-Anforderung

(speziell für Amiga 1000)

- Kickstart 33.180 (Version 1.2) oder höher

Bestell-Nr. 51670

# DM 189,-\*

(sFr 169,-\*/öS 2290,-\*)

\* Unverbindliche Preisempfehlung

**Markt&Technik**  
Zeitschriften · Bücher  
Software · Schulung

Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computer-Fachgeschäften oder in den Fachabteilungen der Warenhäuser.

# Amiga-Software ZING! KEYS

**Definieren Sie individuelle, leistungsstarke Tastatur- und Maus-Makros für alle Anwendungsbereiche.**

Jeder Amiga-Besitzer kennt das: mühselig mechanisch ausgeführte Eingaben von Tastenfolgen zum Aufrufen einer bestimmten, immer wiederkehrenden Funktion. Spazierfahrten mit der Maus kreuz und quer über den ganzen Bildschirm. Zeitverluste bei der Fensterverwaltung. Mit Zing! Keys können Sie all dies und noch viel mehr einfacher und effektiver gestalten. Ein Tastendruck, und die erforderlichen weiteren Eingaben werden automatisch abgearbeitet. Dies gilt auch für Intuition-Funktionen (Fenster öffnen oder schließen, Bewegen des Mauszeigers usw.) und für Betriebssystembefehle. Stellen Sie sich Ihre eigenen, für jede Anwendung verschiedenen Makros zusammen und laden Sie eine spezielle Tastaturbelegung, wenn Sie sie benötigen.

#### Funktionen von Zing! Keys:

- Sicherheitstastatursperre
- Hochleistungstasteneditor
- Speichern von CLI-Kommandofolgen
- Belegungen laden/speichern/zusammenführen
- Makros speichern, auch Verschachtelungen
- Bildschirminhalt als IFF-Grafik speichern
- Verzögerungsfunktion



- Bildschirmabschaltung
- Datums- und Zeitfunktion
- Überspringen von Makrofunktionen
- Fenster vergrößern/verkleinern
- Fenster bewegen
- Fenstergrößen speichern
- Fenster öffnen/schließen
- Fenster in den Vordergrund oder Hintergrund bringen
- Makros vorübergehend stilllegen
- Makros wieder aufnehmen
- Einbindung von Variablen in Makros
- Laufende Makros unterbrechen
- Bildschirminhalt ausdrucken
- Belegungen löschen
- Bildschirm ein- und ausschalten
- automatische Fensteraktivierung
- Speichern von Mausbewegungen

Diese und weitere Funktionen helfen Ihnen dabei, die für Sie passenden Belegungen und Abläufe zu programmieren und zu speichern. Ein unentbehrliches Werkzeug für jeden Amiga-Besitzer!

Bestell-Nr. 51670

**DM 99,-\***

(sFr 89,-\*/öS 990,-\*)

\* Unverbindliche Preisempfehlung

**Markt&Technik**  
Zeitschriften · Bücher  
Software · Schulung

Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computer-Fachgeschäften oder in den Fachabteilungen der Warenhäuser.

Bitte schneiden Sie diesen Coupon aus, und schicken Sie ihn in einem Kuvert an:  
Markt & Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar

# Computerliteratur und Software vom Spezialisten

Vom Einsteigerbuch für den Heim- oder Personalcomputer-Neuling über professionelle Programmierhandbücher bis hin zum Elektronikbuch bieten wir Ihnen interessante und topaktuelle Titel für

• Apple-Computer • Atari-Computer • Commodore 64/128/16/116/Plus 4 • Schneider-Computer • IBM-PC, XT und Kompatible  
sowie zu den Fachbereichen Programmiersprachen • Betriebssysteme (CP/M, MS-DOS, Unix, Z80) • Textverarbeitung • Datenbanksysteme • Tabellenkalkulation • Integrierte Software • Mikroprozessoren • Schulungen.  
Außerdem finden Sie professionelle Spitzen-Programme in unserem preiswerten Software-Angebot für Amiga, Atari ST, Commodore 128, 128 D, 64, 16, für Schneider-Computer und für IBM-PCs und Kompatible!  
Fordern Sie mit dem nebenstehenden Coupon unser neuestes Gesamtverzeichnis und unsere Programmservice-Übersichten an, mithilfe von Utilities, professionellen Anwendungen oder packenden Computerspielen!



Markt & Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2,  
8013 Haar bei München, Telefon (089) 46 13-0

709005

Adresse:

Name

Straße

Ort

Bitte schicken Sie mir:

- Ihr neuestes Gesamtverzeichnis  
 Eine Übersicht Ihres Programmservice-Angebotes aus der Zeitschrift

- Außerdem interessiere ich mich für folgende/n Computer:

(P.S. Wir speichern Ihre Daten und verpflichten uns zur Einhaltung des Bundesdatenschutzgesetzes)

**Markt & Technik Verlag AG**  
**- Unternehmensbereich Buchverlag -**  
**Hans-Pinsel-Straße 2**  
**D-8013 Haar bei München**

# Amiga-Software

# CLimate 1.2

## Jetzt stehen Ihnen die Funktionen Ihres Amiga-Command-Line-Interface per Mausclick zur Verfügung!

Mit diesem Programm können Sie die Befehle des Command-Line-Interface (CLI) benutzerfreundlich und schnell per Mausclick verwenden!

### Ihre Super-Vorteile mit CLimate 1.2:

- sehr große Übersichtlichkeit der Bildschirmdarstellung (Sie haben alle Funktionen auf einen Blick)
- leichte Bedienung aller Befehle mit der Maus
- drei externe Laufwerke (3 1/2" oder 5 1/4"), zwei Festplatten, RAM-Disk unterstützen Sie
- schnelle Directory-Anzeige
- Sie können Disketten leicht nach Texten, Bildern u.ä. durchsuchen
- Dateien lassen sich mit Pause/Continue-Möglichkeit betrachten

- Ausdrucken von Dateien auf Drucker
- Informationen über die Disketten (Programmlänge und ähnliches)
- Betrachten von Bildern im IFF-Format (inklusive HAM)

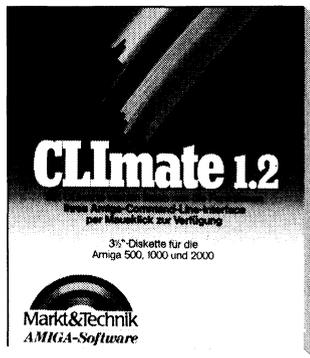
- Sie können Dateien aus beliebigen Verzeichnissen in andere Verzeichnisse kopieren
- Bildschirmausgabe von Dateien in ASCII und in hexadezimaler Form
- Unterstützung von Jokerzeichen bei Disketten- und Dateioperationen

CLimate 1.2 - das unentbehrliche Programm für den Amiga-500-, Amiga-1000- und Amiga-2000-Besitzer.

### Am besten gleich bestellen!

Hardware-Anforderungen: Amiga 500, 1000 oder 2000 mit mindestens 512 Kbyte Hauptspeicher. Empfohlene Hardware: Farbmonitor.

Software-Anforderungen: Kickstart 1.2 (oder ROM bei Amiga 500 und 2000), Workbench 1.2. Eine 3 1/2"-Diskette für den Amiga 500, 1000 und 2000.



Bestell-Nr. 51653

**DM 79,-\***  
(sFr 72,-\*/öS 990,-\*)

\*Unverbindliche Preisempfehlung



Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computer-Fachgeschäften oder in den Fachabteilungen der Warenhäuser.



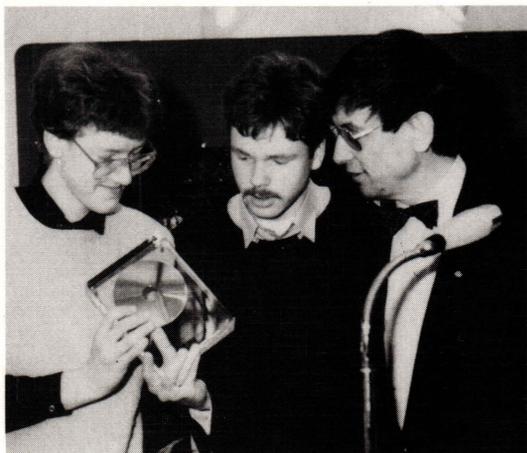
# Amiga Programmier-Handbuch

## Die Autoren:

JÖRG KOCH, geboren 1967, befindet sich zur Zeit in der Ausbildung zum Energie-Anlagen-Elektroniker. Neben der Ausbildung beschäftigt er sich als freier Autor auf dem Spezialgebiet Software-Entwicklung unter anderem mit den Programmiersprachen Modula und C auf verschiedenen 16-Bit-Rechnern.

FRANK KREMSE, geboren 1967, erwarb auf vielen Gebieten der Informatik Grundkenntnisse, die er in Form von Programmen und Berichten in Fachzeitschriften einem breiten Publikum zugänglich gemacht hat.

Beide Autoren haben 1987 mit einem Lernprogramm für den Amiga den Wettbewerb »Goldene Diskette«, unter der Schirmherrschaft von Bundesforschungsminister Dr. Heinz Riesenhuber, erneut gewonnen, nachdem sie bereits 1986 mit einem Programm für den Apple erfolgreich waren.



Bundesforschungsminister Dr. Heinz Riesenhuber übergibt Frank Kremser und Jörg Koch aus Marburg die »Goldene Diskette«.

Der Commodore Amiga bietet ausgezeichnete Voraussetzungen für ein professionelles Entwickeln von Software. Die schnellen Prozessoren garantieren optimale Rechenleistung, und das moderne, grafisch orientierte Betriebssystem öffnen den Weg für eine neue Generation von benutzerfreundlichen Programmen.

Dieses Buch führt Sie in die Interna des Amiga ein. Alle System-Bibliotheken, die das Betriebssystem zur Verfügung stellt, werden ausführlich anhand kleinerer und größerer Beispiele erklärt. Die Programme sind in der offiziellen Commodore-Entwicklersprache Lattice C geschrieben. Ein eigenes Kapitel ist der Installation des C-Compilers gewidmet, was erfahrungsgemäß gerade Einsteigern besondere Schwierig-

keiten bereitet. Um Ihnen das lästige Eintippen der Programme zu ersparen, liegt dem Buch eine 3 1/2"-Diskette mit allen Beispiel-Listings bei.

Die Autoren beschränken sich nicht nur auf eine bloße Aufzählung der Bibliotheks-Routinen. Vielmehr wollen sie aus ihrer Praxis heraus die besonders nützlichen Bibliotheken und Devices vorstellen und das Einbinden in eigenen Programmen zeigen. Auch auf das Umfeld des Amiga wird ausführlich eingegangen. So lernen Sie die wichtigen DOS-Funktionen für das Ansteuern von Druckern oder der SideCar-Harddisk kennen.

## Aus dem Inhalt:

- Die Screen- und Window-Bibliotheken
- Die Zeichen- und Textbefehle
- Hardware-Sprites und Animation
- Aufbau und Abfrage von Menüs
- Gadgets
- Mitteilung von System-Meldungen
- DOS-Funktionen in eigenen Programmen
- Druckerausgabe
- Befehle zur Sprachein- und -ausgabe

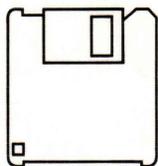
## Hard- und Software-Voraussetzungen:

- Amiga 500, 1000 oder 2000 mit 512 Kbyte RAM-Speicher
- C-Compiler wie Lattice C oder Aztec C

ISB N 3-89090-491-2



Markt & Technik



DM 69,-  
sFr 63,50  
öS 538,20