

330 zusätzliche
Seiten zum Original
Handbuch und mehr als
200 Kommentare zu
Teil A bis C!!!

Für GEOS64 und GEOS128

GEOS-Programmierung
mit dem

Mega

RELOADED!
rev2

ASSEMBLER

Teil D

Mit MegaAssembler V4, GEOS/MegaPatch V3 und
einer Übersicht über alle GEOS-Routinen!

M. Kanet
(w) 1999-2024

GEOS-Programmierung mit dem MegaAssembler »RELOADED«

Ergänzungen zum Original-Handbuch:

- Fehlerkorrekturen und Anmerkungen von 1990 bis 2024
- Erweiterungen für den GEOS/MegaAssembler
- GEOS-Erweiterung "MegaPatch3" für GEOS 64/128

2022 – 2024

Markus Kanet

Version: 25.08.24.rev400-v3

Teil D	Erweiterungen	(ab Seite 431)
---------------	----------------------	-----------------------

	Vorwort	432
Kapitel 1	MegaAssembler V4	433
1.1	Der Menübildschirm	433
1.2	Die neuen Opcodes	437
1.3	Der MegaLinker	442
1.4	Call MegaAss3	443
1.5	Der AutoAssembler	444
1.6	Weitere Änderungen im MegaAssembler	460
1.7	Fehlerkorrekturen	460
1.8	Übersicht über alle Fehlermeldungen	462
1.9	Eintrag in der Labeltabelle	462
Kapitel 2	GEOS/MegaPatch	463
2.0	Speicherbelegung unter GEOS/MegaPatch	463
2.1	Der Speicherbereich von \$8000-\$8fff	464
2.2	Der Speicherbereich von \$9000-\$9d7f	466
2.3	Der Speicherbereich von \$9d80-\$9fff	469
2.4	Farbtabelle für GEOS/MegaPatch	479
2.5	Angepasste Kernalroutinen	480
2.6	Neue Kernalroutinen für GEOS	485
2.7	Das Registermenü	494
2.8	Ausgelagerten Kernalroutinen	508
2.9	Die Laufwerkstreiber für NativeMode	520
2.10	Symboltabellen und Makrodefinitionen	526
Anhang K	Kurzreferenz	538
K.1	Systemübersicht	539
K.2 - K.14	GEOS-Routinen	556
K.15	Das GEOS/MegaPatch Registermenü	646
Anhang L	Die C64-Tastatur unter GEOS	648
L.1	Die Tastaturmatrix des C64	648
L.2	Die Tastaturcodes unter GEOS	649
L.3	Die Tastaturcodes unter GEOS/US	651
Anhang M	Quelltext-Beispiele	652
M.1	GEOS/MegaPatch: Bildschirmschoner "Rasterbars"	652
M.2	GEOS/MegaPatch: Bildschirmschoner "Starfield"	658
M.3	GEOS/MegaPatch: "geoPainter"	672
M.4	Demo/DeskAccessory: "geoScreenCapture"	680
M.5	Demo/Application: "ScrapViewer"	726
M.6	Demo/Application: "keyData"	738
M.7	Demo/Application: "Disk-Analyzer"	742
M.8	Systemroutinen: "EnableIO / DisableIO"	755

Fußnotenverzeichnis

Die folgenden Seitenzahlen beziehen sich auf Fußnoten im Teil A bis C dieses Handbuches und zeigen Hinweise, Korrekturen oder Ergänzungen an.

In einigen wenigen Fällen findet sich die Stelle im Original-Handbuch ein paar Seiten davor oder danach, da in der »**RELOADED**«-Version des Handbuchs das Layout angepasst wurde um zusätzliche Informationen mit aufnehmen zu können.

1. **Seite 38:** Korrektur: Im Original-Handbuch ist "EOR" als Rechenart aufgeführt, wird vom MegaAssembler aber nicht unterstützt.
2. **Seite 43:** Ergänzung: Informationen zum Branch-Fehler im MegaAssembler eingefügt, siehe auch Anmerkungen auf Seite 331.
3. **Seite 44:** Ergänzung: Informationen zum N-Flag bei Vergleichsbefehlen CMP, CPX und CPY ergänzt.
4. **Seite 52:** Ergänzung: Hinweise zum unüberlegten Einsatz von Makros eingefügt (kann den Objektcode unnötig vergrößern).
5. **Seite 53:** Hinweis: Verwendung von Makros kann die Lesbarkeit von Programmcode erschweren.
6. **Seite 59:** Korrektur: Verweise auf "Anhang X" in Kapitel 5.2 und 5.2.1 in "Teil C, Anhang J" geändert.
7. **Seite 60:** Korrektur: Die Größe des Bildschirms wurde im Text angepasst (Höhe/Breite verwechselt).
8. **Seite 67:** Hinweis: Ab GEOS V1.3 ist nicht mehr in jeder Application ein Dolcons-Menü erforderlich.
9. **Seite 67:** Ergänzung: Unter GEOS V1/V2 sucht EnterDesktop nur auf Laufwerk A/B oder C/D nach der Desktop-Datei.
10. **Seite 68:** Hinweis: Ab GEOS V1.3 ist nicht mehr in jeder Application ein Dolcons-Menü erforderlich.
11. **Seite 69:** Korrektur: Für das Schattenmuster einer Dialogbox sind die Bits 0 bis 4 (fünf Bits) zuständig, nicht die ersten vier Bits.
12. **Seite 71:** Korrektur: Die Steuercodes der Routine "PutChar" werden nicht im Anhang IX, sondern im Anhang I.2, aufgeführt.
13. **Seite 71:** Ergänzung: In Dialogboxen kann für die x-Koordinate eines Elements nur ein Wert von 0-255 eingesetzt werden.
14. **Seite 71:** Korrektur: In Bild 5.4 war der falsche Screenshot abgebildet (vgl. Bild 5.6).
15. **Seite 72:** Korrektur: Für die Berechnung der Icon-Position wurden falschen Werte für die Standard-Dialogbox verwendet.
16. **Seite 75:** Ergänzung: Von "InitProcesses" können 20 Prozesse verwaltet werden, Prozessanzahl wird nicht überprüft.
17. **Seite 78:** Ergänzung: Im Gegensatz zu GEOS64 ist unter GEOS128 der I/O-Bereich von \$d000 bis \$dfff immer aktiviert.
18. **Seite 84:** Hinweis: Bei DBGETSTRING entspricht die y-Koordinate der Oberkante der Texteingabe, nicht der Baseline der Zeichen.
19. **Seite 87:** Korrektur: In Bild 5.6 war der falsche Screenshot abgebildet (vgl. Bild 5.4).
20. **Seite 88:** Ergänzung: Es droht Datenverlust in der Hauptanwendung, wenn ein DeskAccessory zum Desktop zurückkehrt.
21. **Seite 88:** Korrektur: Eine evtl. Routine in "appMain" wird erst am Ende eines MainLoop-Durchlaufs aufgerufen.
22. **Seite 89:** Korrektur: Vor dem Laden eines DeskAccessories löscht der GEOS-Kernal die Anzahl der aktiven Prozesse.
23. **Seite 90:** Korrektur: Angaben zur Hintergrundgrafik im 80Z-Modus von "\$a000-\$bf3f" in "\$a040-\$bf7f" im Text angepasst.
24. **Seite 95:** Korrektur: Der Bereich von "COLOR_MATRIX" (Farb-RAM) reicht von \$8c00 bis \$8fe7 (nicht \$8ff7).
25. **Seite 101:** Hinweis: GEOS V2 unterstützt vier Laufwerke, nur Desktop V2 ist auf drei Laufwerke begrenzt.
26. **Seite 101:** Ergänzung: Zweiter Laufwerkstreiber ohne REU nur innerhalb GEOS64 mit Desktop V2 oder GEOS128 möglich.
27. **Seite 111:** Korrektur: Die Routine "GotoFirstMenu" kann auch im Hauptmenü verwendet werden, wird aber sofort beendet.
28. **Seite 113:** Korrektur: Angaben zur Hintergrundgrafik im 40Z-Modus von "\$6000-\$7f39" in "\$6000-\$7f3f" im Text angepasst.
29. **Seite 113:** Korrektur: Angaben zur Hintergrundgrafik im 80Z-Modus von "\$a000-\$bf39" in "\$a040-\$bf7f" im Text angepasst.
30. **Seite 126:** Hinweis: Im Listing_7.1 fehlen vor dem Label "error" ca. 80 Zeilen Code, die nur auf Diskette vorhanden sind.
31. **Seite 147:** Hinweis: Im Listing_8.1 auf Diskette fehlt das File-Icon das hier über den Opcode "i" in den Quelltext eingebunden wird.
32. **Seite 162:** Korrektur: Im Listing_8.4 war eine falsche Grafik für das File-Icon abgebildet.
33. **Seite 162:** Korrektur: Für die Berechnung der Dateigröße erwartet "SaveFile" bei VLIR-Dateien Ladeadresse=Endadresse.

34. **Seite 165:** Korrektur: Der Druckertreiber in GEOS128 liegt im FrontRAM ab \$d9c0 und nicht ab \$df80 im Speicher.
35. **Seite 166:** Korrektur: Auf die Besonderheiten im Zusammenhang mit "GetFile" wird in Kapitel 8.4, nicht 8.5, eingegangen.
36. **Seite 166:** Ergänzung: Im Listing fehlte das setzen des Zeigers in r1 auf den 1920 Byte großen Arbeitsspeicher.
37. **Seite 172:** Ergänzung: Informationen und Beispiele zu Autostart-Programmen unter GEOS eingefügt.
38. **Seite 175:** Ergänzung: Angaben zur Speicherbelegung unter GEOS128 eingefügt.
39. **Seite 177:** Korrektur: "curSetWidth" definiert nicht die Breite eines Zeichens, sondern die Länge einer Bitstream-Reihe.
40. **Seite 177:** Korrektur: Die Adresse "curHeight" wurde hier ursprünglich als "curSetHeight" bezeichnet.
41. **Seite 177:** Korrektur: In "currentMode" müssen für "outline" und "underline" die beiden Bits 3 und 7 (%1000 1000) gesetzt werden.
42. **Seite 178:** Ergänzung: Angaben zu Bit 5 in "dispBufferOn" bei Verwendung von Dialogboxen eingefügt (Bit 5=Dialogbox aktiv).
43. **Seite 179:** Hinweis: Das KEYPRESS_BIT in "pressFlag" sollte durch Anwendungen nicht gelöscht werden (Tastaturpuffer-Problem).
44. **Seite 180:** Ergänzung: Angaben zur Adresse \$003f unter GEOS64 ("Dolcons") hinzugefügt (unter GEOS128 "graphMode").
45. **Seite 181:** Ergänzung: Im Bereich von \$0058-\$006f ist Platz für zusätzliche Zeiger u0 bis u11, wird z.B. von cc65 verwendet.
46. **Seite 181:** Ergänzung: Der Bereich \$0080-\$00fa wird von Anwendungen (geoLinker) und Laufwerkstreibern (\$008b-\$008f) genutzt.
47. **Seite 182:** Korrektur: "kernalVectors" zeigt nicht auf eine Kernaltabelle, sondern hier liegen Vektoren zu Kernalaroutinen.
48. **Seite 182:** Ergänzung: Informationen zu "APP_LVAR" (\$0200-\$0258) und "APP_LRAM" (\$0334-\$03ff) eingefügt.
49. **Seite 182:** Ergänzung: Hinweise zum Speicherbereich \$5000-\$5fff und Autostart-Programmen eingefügt.
50. **Seite 183:** Ergänzung: Informationen zur Hintergrundgrafik im 80Z-Modus und DeskAccessories im Text eingefügt.
51. **Seite 184:** Korrektur: Informationen zu "dir3Head" angepasst und zusätzliche Hinweise zu einer 1581 ohne REU ergänzt.
52. **Seite 185:** Hinweis: Angabe zu "curDrive" (nur Laufwerk A/B) gilt nur für die DeskTop-Oberfläche von GEOS 64/128 V1.x/V2.x
53. **Seite 185:** Korrektur: Das Flag "diskOpenFlg" wird im Original-Handbuch als "diskOpenFlag" bezeichnet.
54. **Seite 185:** Ergänzung: Vom GEOS-Kernal wird "diskOpenFlg" nicht verändert und dürfte für Anwendungen reserviert sein.
55. **Seite 186:** Hinweis: Wert in "numDrives" gilt nur für GEOS V1/V2 mit DeskTop, nicht für andere GEOS-Oberflächen.
56. **Seite 186:** Hinweis: Mit Zusatzanwendungen lässt sich ein viertes Laufwerk in "driveType" einrichten und auch verwenden.
57. **Seite 186:** Hinweis: Laufwerkstyp \$43 (1581 Shadow) wird nicht von allen Configure-Versionen unterstützt.
58. **Seite 186:** Korrektur: "curRecord" enthält nach OpenRecordFile den Wert \$00 (für den ersten Datensatz) oder \$ff (kein Datensatz).
59. **Seite 187:** Korrektur: Angaben zur Adresse "fileWritten" wurden im Text angepasst (\$00=Datensatz bereits gesichert).
60. **Seite 188:** Ergänzung: "mouseVector" enthält normalerweise den Wert \$0000, wird erst durch "StartMouseMode" gesetzt.
61. **Seite 188:** Korrektur: Im Text wurden die Adressen "mouseFaultVector" / "otherPressVector" gem. "TopSym" angepasst.
62. **Seite 188:** Korrektur: Die Adresse "StringFaultVector" wurde im Text gem. "TopSym" angepasst.
63. **Seite 189:** Ergänzung: GEOS128 verwendet für alphanumerische Zeichen zusätzlich Bit 7 und Bit 6 in "c128_alphaFlag" (\$881a).
64. **Seite 191:** Ergänzung: Informationen zu "mousePicData" und Mauszeiger unter GEOS128 im 80-Zeichen-Modus eingefügt.
65. **Seite 192:** Ergänzung: "keyData" wird über die Mainloop, "pressFlag" und "currentKey" über den Interrupt aktualisiert.
66. **Seite 192:** Korrektur: Angaben zur Adresse "mouseData" wurden im Text angepasst (\$80=Feuerknopf nicht gedrückt).
67. **Seite 193:** Hinweis: Adressen "hour", "minutes" und "seconds" für die GEOS-Zeit sind "ReadOnly", Aktualisierung durch Mainloop.
68. **Seite 194:** Ergänzung: Angaben zur internen Belegung von "dlgBoxRamBuf" eingefügt.
69. **Seite 194:** Ergänzung: Angaben zur Systemadresse "currentKey" (\$87ea) eingefügt.
70. **Seite 195:** Ergänzung: Hinweis ergänzt, nicht alle Systemadressen von \$851c bis \$88ff sind unter GEOS V1.2 verfügbar.
71. **Seite 195:** Ergänzung: Angaben zur Adresse "DB_DblBit" (\$8871) und den damit verbundenen Fehlern eingefügt.

72. **Seite 195:** Ergänzung: Informationen zu `":sysRAMFlg"` und `":MoveData"` mit Speicher bis \$3900 unter GEOS128 eingefügt.
73. **Seite 195:** Ergänzung: Angaben zu `":sysRAMFlg"` und zur Lage der Laufwerkstreiber in der REU ab \$8300 eingefügt.
74. **Seite 196:** Ergänzung: Angaben zu `":sysRAMFlg"` und Speicherbelegung von Bank 0 in REU unter GEOS128 eingefügt.
75. **Seite 196:** Ergänzung: GEOS64 testet bei `":MoveData"` mit `":sysRAMFlg"` / Bit 7=1 nicht die Transfergröße (max. \$7900 Byte).
76. **Seite 196:** Korrektur: Angaben zum Register `":firstBoot"` wurden im Text angepasst (\$00=Boot-Vorgang aktiv).
77. **Seite 196:** Ergänzung: Informationen zu `":ramBase"` um den Speicherbedarf von RAM-/Shadow-Laufwerken erweitert.
78. **Seite 197:** Korrektur: In Byte 63 von `":spr0pic"` findet man unter GEOS128 die Höhe des Sprite für den 80-Zeichen-Modus.
79. **Seite 197:** Korrektur: `":obj0Pointer"` ist ein Zeiger auf einen 64-Byte-Block der VIC-Bank, nicht das Lowbyte einer Adresse.
80. **Seite 198:** Ergänzung: Angaben zum Speicherbereich von \$9d80-\$9fff eingefügt.
81. **Seite 198:** Ergänzung: Angaben zum Speicherbereich von \$a040-\$bf7f unter GEOS128 eingefügt.
82. **Seite 198:** Korrektur: Angaben zu `":bootName"` im Text angepasst (Dateiname für Kern-LOAD bei einem Reboot in GEOS v1.2).
83. **Seite 199:** Ergänzung: `":version"` ist erst ab GEOS V1.2 verfügbar, unter GEOS V1.0/V1.1 findet sich hier der Wert \$a9.
84. **Seite 199:** Ergänzung: Weitere GEOS-Sprachversionen für `":nationality"` (\$c00f) aufgelistet.
85. **Seite 199:** Ergänzung: Die Adresse \$c011 ist "Reserved for future use" und wird derzeit nicht verwendet.
86. **Seite 199:** Ergänzung: Angaben zum Register `":geoRamFlg"` (\$c049) eingefügt.
87. **Seite 200:** Ergänzung: Im Gegensatz zu GEOS64 ist unter GEOS128 der I/O-Bereich von \$d000 bis \$dfff immer aktiv.
88. **Seite 201:** Korrektur: Die Adresse `":rasreg"` kann für Rasterzeilen-Interrupts auch beschrieben werden, Angaben ergänzt.
89. **Seite 201:** Ergänzung: Zusätzliche Informationen zum Register `":grmempt"` eingefügt.
90. **Seite 203:** Ergänzung: Angaben zu weiteren Speicherbereichen unter GEOS128 eingefügt.
91. **Seite 204:** Korrektur: Hinweis entfernt, das jede Application einen Aufruf von `":DoMenu"` benötigt.
92. **Seite 204:** Ergänzung: Informationen zum Fehler in `":DoMenu"` beim setzen des Mauszeigers in vertikalen Menüs eingefügt.
93. **Seite 204:** Ergänzung: Informationen zum Fehler in `":DoMenu"` in GEOS128 und x-Koordinaten größer 255 Pixel eingefügt.
94. **Seite 205:** Korrektur: Der Fehler in horizontalen Menüs mit einer x-Koordinate > 255 betrifft nur GEOS128 im 80-Zeichen-Modus..
95. **Seite 207:** Hinweis: Im Gegensatz zu `":Dolcons"` unterstützt `":DoMenu"` kein Sprungziel mit Adresse \$0000.
96. **Seite 210:** Korrektur: Nur bis GEOS V1.2 wird die `":Dolcons"`-Tabelle nicht initialisiert, daher ist ein Dummy-Icon erforderlich.
97. **Seite 210:** Korrektur: `":UN_CONSTRAINED"` steht für ein Menü ohne Mausbegrenzung, nicht für ein selbstabbaubendes Menü.
98. **Seite 211:** Ergänzung: Weitere Informationen zur Verwendung von `":DoPreviousMenu"` in der obersten Menü-Ebene eingefügt.
99. **Seite 211:** Korrektur: Im Text wurde die Schreibweise für `":recoverVector"` an die `":TopSym"` angepasst (`":RecoverVector"`).
100. **Seite 212:** Korrektur: Hinweis entfernt, das jede Application ein `":Dolcons"`-Aufruf benötigt (Nur bis GEOS V1.2 notwendig).
101. **Seite 213:** Ergänzung: `":DoMenu"` unterstützt im Gegensatz zu `":Dolcons"` kein Sprungziel mit Adresse \$0000.
102. **Seite 213:** Korrektur: Nur bis GEOS V1.2 wird die `":Dolcons"`-Tabelle nicht initialisiert, daher ist ein Dummy-Icon erforderlich.
103. **Seite 214:** Korrektur: `":Dolcons"` löscht in `":mouseOn"` das `MENUON_BIT` wenn das `MOUSEON_BIT` ebenfalls gelöscht ist.
104. **Seite 217:** Korrektur: Das Schattenmuster der Dialogbox wird in den ersten fünf Bit (0-4) definiert, nicht in den ersten vier Bit.
105. **Seite 219:** Korrektur: Im Original-Handbuch wird der Funktionscode `":DBOPVEC"` als `":DBSOPV"` bezeichnet.
106. **Seite 219:** Korrektur: Der Funktionscode `":DBUSRICON"` wird im Original-Handbuch als `":DBUSERICON"` bezeichnet.
107. **Seite 221:** Korrektur: Im Original-Handbuch wurde hier die Routine `":RstrFrmDialogue"` als `":RstrFrmDialog"` bezeichnet.
108. **Seite 221:** Korrektur: Im Text wurde die Schreibweise für `":recoverVector"` an die `":TopSym"` angepasst (`":RecoverVector"`).
109. **Seite 222:** Ergänzung: Fehlerbeschreibung zu GEOS128 und Dialogboxen mit System-Icons im 80Z-Modus eingefügt.
110. **Seite 222:** Ergänzung: Fehlerbeschreibung zu GEOS128 und Dialogboxen ohne Schatten im 80Z-Modus eingefügt.

111. **Seite 223:** Korrektur: Lage der Hintergrundgrafik im 802-Modus angepasst ("SCREEN_BASE"+\$40).
112. **Seite 226:** Korrektur: Im Gegensatz zur Routine "Horizontalline" verändert "RecoverLine" nicht das Register r11h.
113. **Seite 226:** Korrektur: Die Routine "InvertLine" arbeitet ähnlich wie "Horizontalline", verändert aber nicht r7 und r11h.
114. **Seite 228:** Ergänzung: Abgaben zu "GetScanLine" und den veränderten Registern eingefügt.
115. **Seite 228:** Ergänzung: Begründung zu "SetPattern" und den maximal unterstützten Muster 0-31 eingefügt.
116. **Seite 234:** Korrektur: In "GraphicsString" wurden die Konstanten "PEN...DELTA" im Text an die Datei "TopSym" angepasst.
117. **Seite 242:** Korrektur: Hinweis in "BitOtherClip" auf "DOUBLE_W" in "DOUBLE_B" für Bytewerte in r1L/r2L geändert.
118. **Seite 242:** Ergänzung: Informationen über zusätzliche GEOS128-Routinen eingefügt.
119. **Seite 242:** Hinweis: "NormalizeX" funktioniert auf Grund eines Fehlers nicht mit negativen Zahlen.
120. **Seite 243:** Korrektur: Text wurde angepasst, da "NormalizeX" nur auf 16-Bit-Werte (Word) angewendet werden kann.
121. **Seite 244:** Ergänzung: Routine "PutChar" testet Zeichen nicht auf Gültigkeit, nur Zeichen zwischen 8 und 127(128) möglich.
122. **Seite 244:** Ergänzung: Die Verwendung von ungültigen Steuercodes kann einen Panic-Systemfehler verursachen.
123. **Seite 245:** Korrektur: In "PutChar" wurde die Beschreibung zu FORWARDSPACE angepasst (ohne Funktion).
124. **Seite 245:** Korrektur: HOME setzt in "PutChar" die x-ly-Koordinate auf Null, nicht auf die erste mögliche Ausgabeposition.
125. **Seite 246:** Ergänzung: "SmallPutChar" unterstützt keine automatische Verdopplung der x-Koordinate über DOUBLE_W/ADD1_W.
126. **Seite 246:** Korrektur: Das Clipping in "SmallPutChar" funktioniert nur am linken Bildschirmrand, nicht am linken Fensterrand.
127. **Seite 247:** Ergänzung: Die Routine "PutString" verändert weitere Register, Angaben zu r0, r1h und r1l eingefügt.
128. **Seite 251:** Ergänzung: Angaben zu "CmpString" und zum y-Register bei unterschiedlichen Strings ergänzt.
129. **Seite 251:** Korrektur: Das Ergebnis von "CmpFString" wird nicht in STATUS übergeben, sondern wie bei "CmpString".
130. **Seite 251:** Korrektur: Bei überlappenden Strings in "CopyString" muss der Ursprungstext hinter dem Zieltext liegen.
131. **Seite 253:** Ergänzung: Informationen zur Routine "InitTextPrompt" eingefügt.
132. **Seite 257:** Ergänzung: Informationen zur Routine "SetMsePic" und veränderte Register eingefügt.
133. **Seite 257:** Korrektur: Die Routine "TempHideMouse" verändert auch das y-Register.
134. **Seite 260:** Ergänzung: Informationen zu "PosSprite" und der Verwendung von "DOUBLE_W" und "ADD1_W" eingefügt.
135. **Seite 260:** Ergänzung: Beschreibung zur Routine "DoSoftSprites" für GEOS128 ergänzt.
136. **Seite 262:** Hinweis: Informationen zur Speicherverwaltung der REU unter GEOS ergänzt.
137. **Seite 263:** Ergänzung: Die Routine "InitRam" darf nicht zur Initialisierung von r0-r2L verwendet werden.
138. **Seite 263:** Korrektur: Mit "MoveData" können bis 64K verschoben werden, mit REU-DMA \$7900(C64) bzw. \$3900(C128).
139. **Seite 264:** Korrektur: MoveData arbeitet nur im FrontRAM (Bank 1). Bezeichnungen der beiden Speicherbänke angepasst.
140. **Seite 264:** Hinweis: Für "DoBop" mit y-Register = %01 gibt es keine passende GEOS-Routine, die Funktion ist zudem fehlerhaft.
141. **Seite 264:** Korrektur: Angabe von TRUE/FALSE bei "DoBop" angepasst, \$ff steht für nicht identische Speicherbereiche.
142. **Seite 265:** Korrektur: Angaben bei "VerifyBDATA" angepasst, \$ff steht für nicht identische Speicherbereiche.
143. **Seite 265:** Korrektur: Die Routine "DoRAMOp" verändert auch den Inhalt von Akku und x-Register.
144. **Seite 266:** Ergänzung: Angaben zu "VerifyRAM" und Bit 5=Ergebnis des Vergleichs eingefügt.
145. **Seite 266:** Korrektur: Bezeichnung der "DoRAMOp"-Routinen an die Symbolnamen in "TopSym" angepasst.
146. **Seite 273:** Ergänzung: Prozess-Status NOTIMER nicht implementiert ("Official GEOS Programmers Reference Guide").
147. **Seite 275:** Korrektur: "RestartProcess" löscht FROZEN_BIT (5), BLOCKED_BIT (6), nicht aber das RUNABLE_BIT (7).
148. **Seite 275:** Ergänzung: "EnableProcess" könnte für NOTIMER-Prozesse genutzt werden, ist aber nicht implementiert.
149. **Seite 279:** Ergänzung: Warnhinweis zur Verschachtelung von "InitForIO" und "DoneWithIO" eingefügt.
150. **Seite 281:** Korrektur: Die Routinen "ExitTurbo" und "PurgeTurbo" übergeben im x-Register keine Fehlermeldung.
151. **Seite 282:** Korrektur: Die Routine "SetGEOSDisk" verändert durch "SetNextFree" auch das Register r3.
152. **Seite 283:** Ergänzung: Informationen zu "GetPtrCurDkNm" ab GEOS V1.3 und Laufwerk C/D eingefügt.

153. **Seite 283:** Ergänzung: In GEOS V1.2 werden von "GetPtrCurDkNm" weitere Register verändert (Aufruf von "BBMult").
154. **Seite 283:** Korrektur: Die Routine "OpenDisk" verändert r0 bei Shadow-Laufwerken, "driveType" wird nicht verändert.
155. **Seite 285:** Ergänzung: In "dirEntryBuf" und "fileHeader" übergibt "GetFile" weitere Informationen zur geladenen Datei.
156. **Seite 286:** Ergänzung: Informationen zu "SaveFile" und der Endadresse+1 bei sequentiellen Dateien eingefügt.
157. **Seite 286:** Ergänzung: Hinweis zu "SaveFile" und Infoblock eingefügt (Infotext wird gelöscht).
158. **Seite 287:** Ergänzung: Zusätzliche Hinweise zu "RstrAppl" und zum Beenden eines DeskAccessory eingefügt.
159. **Seite 287:** Korrektur: Die Routine "NewDisk" verändert r1, bei einem Shadow-Laufwerk zusätzlich noch r0, r2 und r3.
160. **Seite 288:** Korrektur: Die Routine "CalcBlksFree" übergibt im x-Register keine Fehlermeldung.
161. **Seite 288:** Ergänzung: Hinweis zu "GetDirHead" ohne "NewDisk" und \$29.DSK_ID_MISMAT bei 1541/71-Laufwerken eingefügt.
162. **Seite 288:** Korrektur: "GetDirHead"/"PutDirHead" liefern ab GEOS V1.5 in r4 einen Zeiger auf verschiedene Speicherbereiche.
163. **Seite 291:** Korrektur: Im 1541-Laufwerkstreiber existiert "AllocateBlock" ab GEOS V1.5 mit KONFIGURIEREN V1.6.
164. **Seite 292:** Korrektur: Die Routine "FreeFile" verändert auch den Speicherbereich ab "diskBlkBuf".
165. **Seite 293:** Ergänzung: "GetFreeDirBlk" übergibt in r4 zusätzlich einen Zeiger auf "diskBlkBuf".
166. **Seite 293:** Hinweis: "SetGDirEntry" erwartet in r6 zwingend einen Zeiger auf "fileTrScTab".
167. **Seite 293:** Ergänzung: Informationen zur Routine "SetGDirEntry" und Infoblock eingefügt.
168. **Seite 294:** Hinweis: "BldGDirEntry" erwartet in r6 zwingend einen Zeiger auf "fileTrScTab".
169. **Seite 295:** Ergänzung: "GetNxtDirEntry" übergibt im y-Register einen Wert <0 wenn das Verzeichnissende erreicht wurde.
170. **Seite 295:** Ergänzung: Informationen zur Routine "GetNxtDirEntry" und GEOS-Disketten / BorderBlock eingefügt.
171. **Seite 295:** Ergänzung: Informationen zur Routine "FollowChain" und der Puffergröße eingefügt.
172. **Seite 296:** Korrektur: Die Routine "FastDelFile" nutzt intern "FindFile" und verändert daher zusätzliche Register.
173. **Seite 297:** Hinweis: "LdFile" kann nicht direkt verwendet werden, da wichtige Parameter nicht übergeben werden können.
174. **Seite 298:** Korrektur: Information zur Routine "ReadFile" und Sektorverkeftung/Anzahl Byte im letzten Block angepasst.
175. **Seite 299:** Hinweis: Die Routine "ChangeDiskDevice" ist bereits im Original-Handbuch doppelt aufgeführt.
176. **Seite 299:** Ergänzung: Die Routine "ReadLink" erfordert den Aufruf von "EnterTurbo", "InitForIO" und "DoneWithIO".
177. **Seite 299:** Korrektur: Im 1541-Laufwerkstreiber existiert "ReadLink" ab GEOS V1.5 mit KONFIGURIEREN V1.6.
178. **Seite 301:** Korrektur: Die Beschreibung der Routine "PutBlock" wurde im Text angepasst (vermutl. Copy&Paste-Fehler).
179. **Seite 301:** Korrektur: Die Routine "VerWriteBlock" sollte wegen "InitForIO" nicht mit "PutBlock" kombiniert werden.
180. **Seite 301:** Ergänzung: Im 1581-Laufwerkstreiber ist die Routine "VerWriteBlock" ohne Funktion.
181. **Seite 302:** Ergänzung: Informationen zu "GetDiskBlkBuf", "PutDiskBlkBuf", "GetOPDPtr" und "AccessCache" eingefügt.
182. **Seite 303:** Korrektur: "OpenRecordFile" liefert in r1L/r1H nicht den Directory-Block zurück, Angaben korrigiert.
183. **Seite 304:** Korrektur: Entgegen verschiedener Errata zum Handbuch verändert "PointRecord" r4 nicht.
184. **Seite 304:** Korrektur: Die Routine "AppendRecord" verändert zusätzlich auch die Register r1 und r4.
185. **Seite 306:** Korrektur: Angaben zur Lage des Drucktreibers im FrontRAM ab \$d9c0 von GEOS128 im Text angepasst.
186. **Seite 306:** Ergänzung: Informationen zum Fehler in GeoCalc mit größeren Drucktreibern eingefügt.
187. **Seite 309:** Korrektur: Die Routine "SetNLQ" erfordert nicht zwingend den Aufruf von "StartASCII".
188. **Seite 309:** Ergänzung: Informationen zur Routine "PrintFCodes" eingefügt.
189. **Seite 310:** Korrektur: An Stelle von "intTopVector" sollte man "intBotVector" für eigene Interrupt-Aufgaben verwenden.
190. **Seite 311:** Korrektur: Adresse der Routine "DolnlineReturn" und die Anzahl der Inlinebytes wurden im Text angepasst.
191. **Seite 311:** Ergänzung: In GEOS V2 sucht "EnterDeskTop" entweder auf Laufwerk A:/B: oder C:/D: nach DeskTop.
192. **Seite 311:** Hinweis: Es gibt Patches für GEOS, die einen Ausweg aus der Panic-Box erlauben.
193. **Seite 312:** Ergänzung: Angaben zu Speicherbereichen für "BootGeos" unter GEOS128 im Text eingefügt.
194. **Seite 312:** Ergänzung: Angaben zur Funktionsweise von "BootGeos" hinzugefügt.

195. **Seite 312:** Korrektur: Informationen zur Routine "ToBasic" und Register r7 = Programmende im Text korrigiert.
196. **Seite 312:** Hinweis: Angaben zur Routine "ToBasic" und der Zeichenart für BASIC-Befehle ergänzt.
197. **Seite 313:** Ergänzung: Zwei Beispielprogramme für die Routine "ToBasic" im Text ergänzt.
198. **Seite 313:** Ergänzung: Informationen zur Routine "GetSerialNumber" eingefügt.
199. **Seite 321:** Ergänzung: MegaAssembler ersetzt Tabulatoren in Strings durch Leerzeichen.
200. **Seite 321:** Korrektur: MegaAssembler unterstützt auch GEOS-Klassen mit 6 Zeichen in der Form "Vw.xyz".
201. **Seite 327:** Korrektur: Der Assembliervorgang lässt sich nicht über RUN/STOP abbrechen.
202. **Seite 330:** Ergänzung: Informationen zum MegaAssembler und Fehler bei einem Überlauf des Symbolspeicher eingefügt.
203. **Seite 331:** Korrektur: Branch-Fehler im MegaAssembler. Im Text wurden die Werte von -128 und +127 Bytes angepasst.
204. **Seite 332:** Hinweis: Informationen zum MegaAssembler und Strings mit einer Länge >1 als Zahlenangabe ergänzt.
205. **Seite 336:** Hinweis: Information zur Speichergrenze eingefügt, wird von JMP (\$adr) nicht unterstützt.
206. **Seite 347:** Hinweis: Information zur Speichergrenze eingefügt, wird von JMP (\$adr) nicht unterstützt.
207. **Seite 357:** Ergänzung: "V-Link" unterstützt, im Gegensatz zum MegaAssembler, keine bedingte Assemblierung.
208. **Seite 358:** Hinweis: Mit dem Original "V-Link" sind keine GEOS-Klassen in der Form "Vx.yz" oder "Vw.xyz" möglich.
209. **Seite 392:** Ergänzung: Informationen zum CBM-Bootsektor beim konvertieren einer GEOS-Diskette unter DeskTop V2 eingefügt.
210. **Seite 393:** Ergänzung: Hinweise zum DeskTop V2 und schreiben der GEOS-Seriennummer auf Hauptdisketten eingefügt.
211. **Seite 394:** Ergänzung: Angaben zum Programm "GEOS TOOLS" (Disk-Analyzer) und dessen Autor im Text eingefügt.
212. **Seite 395:** Korrektur: Die 1581 kann 296(288) Dateien im Verzeichnis speichern, DeskTop V2 zeigt nur 144 Dateien an.
213. **Seite 398:** Korrektur: Im Infoblock ist eine Versionsangabe ab Byte 89 in der Form "Vx.yz" oder "Vw.xyz" möglich.
214. **Seite 399:** Ergänzung: Bei Zeichensätzen findet sich im Infoblock in den Bytes 97-159 Angaben zu den Punktgrößen.
215. **Seite 399:** Ergänzung: In Byte 134 im Infoblock von GeoPaint-V1.1-Dokumenten steht der Farbmodus, \$00=Farbe aus.
216. **Seite 400:** Ergänzung: Der Indexblock einer VLIR-Datei mit genau 127 Datensätzen hat keine Ende-Kennung mit \$00,\$00.
217. **Seite 403:** Korrektur: Text angepasst, da ein GeoWrite-Dokument nur 63 Bilder aufnehmen kann (Datensatz 64-126).
218. **Seite 406:** Ergänzung: Informationen zu PrintText und Farbcodes eingefügt.
219. **Seite 407:** Ergänzung: Angaben zum Format eines Textscrap eingefügt.
220. **Seite 407:** Ergänzung: Angaben zum Aufbau des VLIR-Header von GeoPublish-Dokumenten ergänzt.
221. **Seite 409:** Ergänzung: Zusätzliche Angaben zur Hintergrundgrafik im 80Z-Modus eingefügt.
222. **Seite 412:** Ergänzung: Im GeoPaint-Format muss ein Datensatz mit einem zusätzlichen NULL-Byte abgeschlossen werden.
223. **Seite 412:** Ergänzung: Angaben zum Format eines Photoscrap ergänzt.
224. **Seite 414:** Korrektur: In der Laderoutine wird in r2 für die Größe des Puffers definiert, nicht die Zeichensatzlänge.
225. **Seite 415:** Korrektur: Angaben zu Fonts aus dem offiziellen GEOS-Programmierer-Handbuch ergänzt.
226. **Seite 415:** Korrektur: Die max. Anzahl an Punktgrößen in einem Font wurde von 12 auf 15 im Text angepasst.
227. **Seite 423:** Korrektur: Die Füllmuster 34/35 im Original-Handbuch gibt es nicht.

GEOS-Programmierung mit dem MegaAssembler »**RELOADED**«

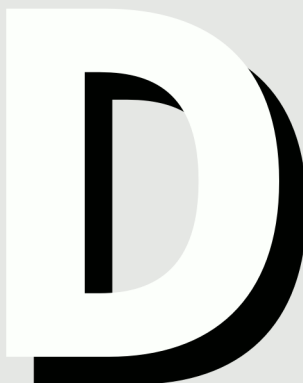
TEIL D

Ergänzungen und Erweiterungen

GEOS/MegaAssembler V4

GEOS/MegaPatch V3

Anhang K - M



Vorwort

Seit dem Erscheinen von "GEOS-Programmierung mit dem MegaAssembler" sind aktuell über 30 Jahre vergangen.

In dieser Zeit wurde aber nicht nur der MegaAssembler überarbeitet, auch das GEOS-Betriebssystem von Berkeley Softworks wurde mit Hilfe von Software-Anpassungen und neuer Hardware erweitert.

MegaAssembler

Mit dem MegaAssembler lassen sich nicht nur kleinere GEOS-Projekte erstellen, auch größere Programme können damit umgesetzt werden.

Das wurde bereits Mitte der 1990er-Jahre möglich, als für GEOS neue Hardware verfügbar war, die deutlich mehr Speicher als ein 1581-Laufwerk oder eine RAM-Erweiterung vom Typ Commodore 1750 zur Verfügung stellte.

Bei größeren Projekten kommt der MegaAssembler allerdings sehr schnell an seine Grenzen, z.B. was die Anzahl der zu assemblierenden Quelltexte oder die Größe der Quelltext-Dateien angeht.

Als GEOS damals die neuen Laufwerke vom Typ "Native-Mode" unterstützte, konnte man damit bis zu 16Mb an Speicher für Quelltexte nutzen. Der MegaAssembler kann aber nur die ersten 13 GeoWrite-Textdateien im aktuellen Laufwerk über das Menü »Texte« anzeigen, egal wie groß das Laufwerk ist.

Das Ergebnis war damals, das man entweder die Texte in eine Vielzahl an Unterverzeichnissen sortieren musste oder immer 13 Dateien auf eine RAM-Disk kopiert und diese dann abwechselnd assembliert. Das alles wurde, mit den immer größer werdenden Projekten, sehr zeitaufwändig und es war notwendig den MegaAssembler zu erweitern.

Damit das möglich wurde musste der Programmcode des MegaAssembler wieder in einen Quelltext umgewandelt werden, da der Original-Quelltext zum Programm zum damaligen Zeitpunkt nicht zur Verfügung stand.

Nach und nach wurde MegaAssembler dann erweitert und schon damals die ersten bekannten Fehler im Programm korrigiert. Die Verbesserungen wurden allerdings erst 20 Jahre später an andere GEOS-Programmierer weitergegeben.

Im **KAPITEL 1** wird davon ausgegangen, das man mit dem Umgang des MegaAssembler vertraut ist. Es werden daher nur die Neuerungen beschrieben.

GEOS

Auch für das GEOS-Betriebssystem gab es in den 1990er-Jahren bereits eine Vielzahl an Erweiterungen und Patches, aber es war keine komplette Lösung. Je nach Hardware wurden auch weiterhin angepasste Startdisketten benötigt. Es bedurfte daher einer Lösung, die mit dem Flickenteppich an Patches aufräumt.

Dazu war es ebenfalls erforderlich das GEOS-Betriebssystem wieder in einen Quelltext umzuwandeln, damit man Erweiterungen ergänzen konnte. Neben dem GEOS-Kernal wurden auch die Laufwerkstreiber überarbeitet. Das Ergebnis der Überarbeitungen war dann der GEOS/MegaPatch.

Im **KAPITEL 2** werden daher die neuen Routinen, Register und Funktionen der GEOS/MegaPatch-Erweiterung besprochen.

KAPITEL 1

Der MegaAssembler V4

MegaAssembler V4 (im Folgenden nur noch als **MA4** bezeichnet) bietet sowohl an der Oberfläche als auch innerhalb des Programms gegenüber der ursprünglichen Version MegaAssembler V2 (im Folgenden nur noch als **MA2** bezeichnet) viele neue Möglichkeiten. Diese werden nun im Detail beschrieben.

1.1 Der Menübildschirm

Bereits nach dem ersten Start fallen hier die ersten Änderungen auf, daher zuerst ein Bild des neuen Startbildschirms.

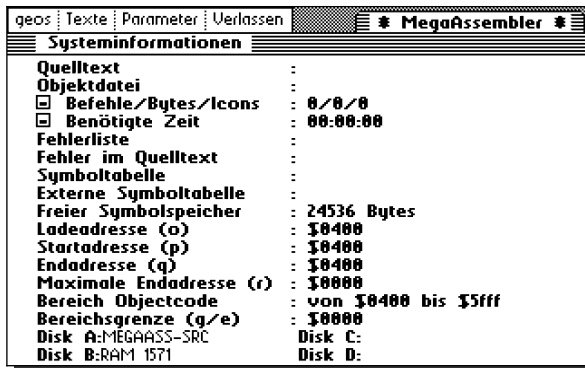


Bild 1.1: Die Oberfläche von MegaAssembler V4

Die ersten beiden Zeilen kennt man noch von der Version **MA2**: »Quelltext« bezeichnet den zuletzt ausgewählten Quelltext und »Objektdatei« die zuletzt assemblierte bzw. erzeugte Objektdatei.

Die beiden nächsten Zeilen dienen nur der Statistik und geben Informationen zur Anzahl der verarbeiteten Befehle im Quelltext, der Anzahl der erzeugten Bytes in den erstellten Objektdateien und die Anzahl der Grafiken im Quelltext. Die Zähler werden nicht automatisch zurückgesetzt, sondern addieren die Angaben während man weitere Quelltexte assembliert.

Das gilt auch für die benötigte Zeit. Größere Projekte können durchaus mehrere Stunden benötigen, bis alle Quelltexte verarbeitet wurden. Der Wert hier kann auch als eine Art "Benchmark" verwendet werden um die benötigte Assemblierungszeit auf verschiedenen Systemen zu testen. Für die weitere Arbeit mit dem **MA4** sind diese beiden Informationen aber nicht weiter von Bedeutung.

Die nächsten vier Zeilen kennt man so auch vom **MA2**. Diese zeigen den Namen der erzeugten Fehlerdatei, die Anzahl der gefundenen Fehler, den Namen der Symboldatei und der externen Symboldatei.

Hier werden nur Daten angezeigt, wenn die entsprechenden Dateien auch erstellt wurden, ansonsten bleiben diese Felder leer.

Bei der Angabe des »Freien Symbolspeichers« fällt die erste Erweiterung auf: Die Zahl wurde beim **MA4** gegenüber **MA2** um ca. 50% vergrößert: Es stehen jetzt knapp 24Kb für Symbole, Labels und Makros zur Verfügung!

Gerade bei größeren Projekten kann es mit der älteren Version schnell zu Problemen kommen und man bekommt den Fehler "Speicher für Symbole/Makros ist voll!" angezeigt. In dem Fall muss man die Anzahl oder die Länge der Labels bzw. die Include-Dateien (z.B. "TopSym" oder "TopMac") reduzieren.

Die nächsten vier Zeilen sollten auch bekannt sein: Die »Startadresse« und die »Ladeadresse« werden durch die entsprechenden Opcodes im Quelltext definiert.

Bei der Endadresse werden nun zwei Werte angezeigt: Zum einen die Adresse des zuletzt assemblierten Byte bzw. der benutzerdefinierten Endadresse über den Opcode **q**, und zum anderen die über den Opcode **r** definierte Endadresse.

Die beiden Werte müssen nicht identisch sein, z.B. kann bei einem DeskAccessory ein größerer Programmbereich als erforderlich definiert werden, damit GEOS den zusätzlichen Speicher beim Start des DeskAccessory auch im Swapfile auslagert. Der Opcode **r** hingegen definiert wie groß das Programm maximal sein darf.

»Bereich Objektcode« definiert den durch das erzeugte Objektfile benötigten Speicherbereich. Hier sind z.B. Werte von \$0400 (**APP_RAM**) bis \$ffff möglich.

»Bereichsgrenze« definiert feste bzw. max. erlaubte Bereichsgrenzen für den Objektcode. Auf diese Neuerung gehen wir später noch im Detail ein.

Darunter werden dann noch die aktuellen Namen der Disketten in den Laufwerken A: bis D: angezeigt. Mit dem Programm "InstallDriveD" ließ sich damals bereits ein viertes Laufwerk unter GEOS nutzen. Allerdings waren dazu andere Programmoberflächen als DeskTop erforderlich. **MA4** unterstützt bei der Laufwerksauswahl aber direkt alle installierten Laufwerke.

Das Menü »Texte«

Eine der großen Nachteile des **MA2** war, dass im Menü »Texte« nur die ersten 13 Quelltexte angezeigt wurden. Bei größeren Projekten muss man also entweder Dateien umsortieren oder mit verschiedenen Disketten arbeiten.

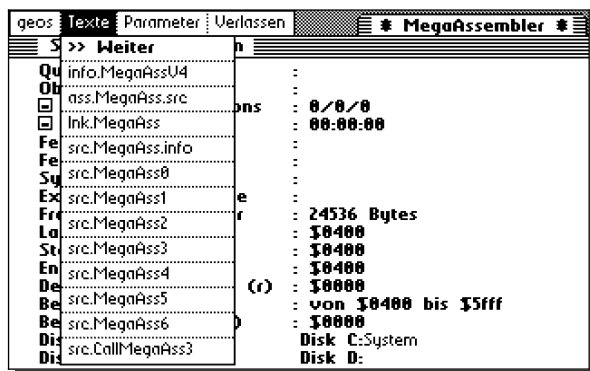


Bild 1.2: MegaAssembler
»Texte«-Menü

MA4 zeigt wie bisher auch die ersten 13 Quelltexte im Menü an, kann aber auch mit mehr als 13 Dateien umgehen.

Befinden sich mehr als 13 Quelltexte auf der Diskette, dann wird als erster Eintrag **>>Weiter** zusammen mit den ersten 12 Quelltexten im Menü angezeigt.

Mit einem Mausklick auf **>>Weiter** werden dann jeweils die nächsten 12 Quelltexte im »Texte«-Menü angezeigt. Wenn das Ende des Verzeichnisses erreicht wurde bzw. weniger als 12 Quelltexte angezeigt werden, dann wird der erste Eintrag auf **<<Anfang** geändert und ein Mausklick springt an den Anfang des Verzeichnisses zurück.

Die Anzeige ist allerdings auf max. 144 Dateien begrenzt (entspricht der max. Anzahl an Dateien auf einem 1541-Laufwerk).

Das Menü »Parameter«

Auch im Menü »Parameter« gibt es neue Funktionen:

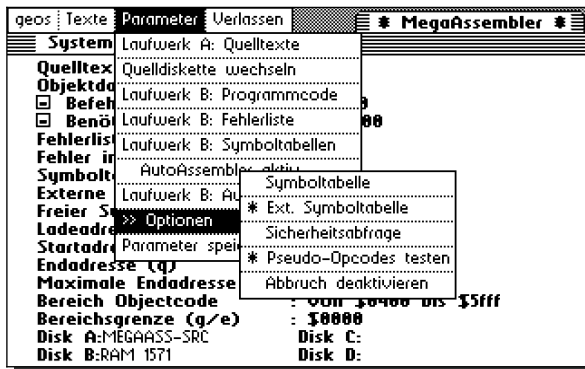


Bild 1.3: MegaAssembler
»Parameter«-Menü

Das Laufwerk für die Quelltexte und die Programm-Dateien lassen sich getrennt voneinander konfigurieren. Auch das Ausgabelaufwerk für die Fehlerliste und für die (externen) Symboltabellen lassen sich separat einstellen.

Danach folgen die Parameter »AutoAssembler« und »Laufwerk AutoAssembler«. Diese Parameter steuern eine weitere Neuerung im **MA4**: Das automatische Assemblieren von mehreren Quelltext-Dateien ohne Benutzereingabe. Wie das im Detail funktioniert wird später noch erklärt.

Die Optionen »Symboltabelle« und »Ext. Symboltabelle« wurden vom **MA2** unverändert übernommen und steuern die Ausgabe von Symboltabellen bzw. von externen Symboltabellen (z.B. für die Verwendung in VLIR-Modulen).

Die »Sicherheitsabfrage« warnt vor dem Überschreiben einer Objektcode-Datei.

Neu hingegen ist die Option »Pseudo-Opcodes testen«. Verwendet man z.B. einen Opcode mit einem nicht definierten Label als Parameter, dann findet bisher keine Fehlerprüfung statt. Das ist auch die Vorgabe wenn man den **MA4** das erste Mal startet.

Aktiviert man aber diese Einstellung, dann wird im 1.Pass geprüft, ob das Label definiert wurde und ggf. ein Fehler "Label unbekannt" erzeugt.

Ist die Option »Abbruch deaktivieren« aktiviert (*), dann lässt sich der Vorgang des Assemblieren nicht mehr per Tastendruck oder Mausklick abbrechen.

Gegenüber **MA2** ist in diesem Menü nur die Option »Sofortstart« weggefallen, da die Assemblierung immer sofort gestartet wird wenn ein Quelltext ausgewählt wird.

Das Menü »Verlassen«

Das Menü »Verlassen« wurde ebenfalls erweitert. Die größte Änderung ist, dass der V-Link jetzt nicht mehr als eigenständiges Programm existiert, sondern direkt in den MegaAssembler integriert wurde.

Über das Menü »Verlassen | MegaLinker« kann man den Linker direkt aufrufen. Man kann nach dem Linken auch wieder zum MegaAssembler zurückkehren, z.B. um weitere Quelltexte zu assemblieren.

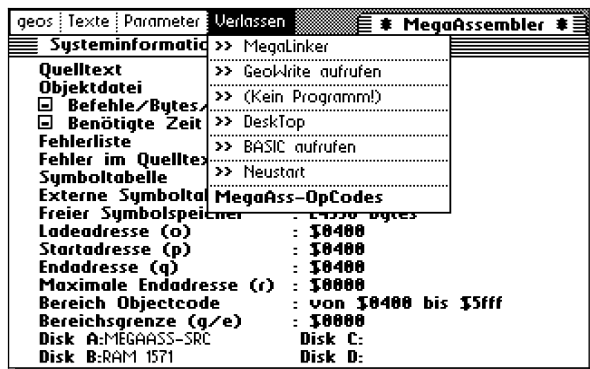


Bild 1.4: MegaAssembler
»Verlassen«-Menü

Der Menüpunkt »GeoWrite aufrufen« ist nahezu unverändert. Hier kann man den Quelltext öffnen, die Fehlerliste oder die Symboltabellen öffnen und außerdem kann man GeoWrite auch ohne Textdatei starten.

In der nächsten Zeile steht der Name der zuletzt assemblierten Datei (bzw. »(Kein Programm)« wenn noch keine Application assembliert wurde.

Neben der Rückkehr zum »Desktop« kann man jetzt auch in das »BASIC« des C64 wechseln, z.B. wenn man MegaAssembler verwendet hat um ein Programm für den BASIC-Modus zu assemblieren. Dabei sollte man aber beachten dass man die assemblierte Programmdatei auf einem realen Diskettenlaufwerk speichern muss, da man von BASIC aus nicht auf ein RAM-Laufwerk von GEOS zugreifen kann.

Für die Opcodes wurde noch noch eine Übersichtsseite ergänzt:

***** MegaAssembler - OpCodes ***** (* = NEU)		
a,c,n	'NAME'	Autor/Klasse/Name festlegen.
f	TYPE	GEOS-Dateityp festlegen.
* h	'TEXT'	Text für Infoblock festlegen.
i	ICON	Infoblock-Icon für Objektdatei.
o,p,q	\$XXXX	Lade-/Start-/Endadr. festlegen.
z	\$XX	Bildschirm-Modus festlegen.
b	\$XX,'TEXT'	Byte-Tabelle einbinden.
s	\$XX	Anzahl \$00-Bytes einbinden.
w	\$XXXX	WORD-Tabelle einbinden.
j	ICON	Infoblock-Icon einbinden.
d	'NAME'	Seq. Datei einbinden.
t	'NAME'	Textdatei einbinden.
v,u	NR,'NAME'	ULIR-Datensatz/Foto einbinden.
* e,q	\$XXXX	Adr. auf Überschreitung testen und bei 'e' mit \$00-Bytes auffüllen.
* r	\$XXXX	Max. Programm-Endadresse festlegen.
* k,l		Datum (kurz/lang) einbinden.
* x,y		Zeit (kurz/lang) einbinden.

Bild 1.5: MegaAssembler
Opcode-Übersicht

Die Opcodes, die auf der Übersichtsseite mit einem * markiert sind, gibt es nur in **MA4**. Was diese neuen Opcodes bedeuten wird im folgenden Abschnitt erklärt.

1.2 Die neuen Pseudo-Opcodes

Um das assemblieren von größeren Projekten weiter zu vereinfachen wurden auch neue Pseudo-Opcodes in **MA4** integriert. Dabei ist zu beachten das Quelltexte, die Pseudo-Opcodes für den **MA4** enthalten, nicht mehr unter **MA2** assembliert werden können! **MA2** würde einen solchen Versuch mit dem Fehler "Befehl unbekannt" quittieren.

h "Text..."

Über den Pseudo-Opcode **h** kann man für den Infotext im Infoblock der Objektdatei einen Text vorgeben. Mit Hilfe der bedingten Assemblierung kann man so z.B. auch Infotexte für eine deutsche und eine englische Version im den Quelltext integrieren. Hier ein Beispiel:

```
if .p
    t "TopSym"
    t "TopMac"
    t "LANG.ext"           ;Enthält Sprachvorgabe
endif
    n "TESTPROG"
    o $010e
    z $80

; Infotext für Infoblock definieren.
if Sprache = Deutsch
    h "Testprogramm"
    h "Für GE0S64 und GE0S128..."
endif

if Sprache = Englisch
    h "Test application"
    h "For GE0S64 and GE0S128..."
endif
endif
```

In der externen Symboldatei "LANG.ext" wird die Konstante »:Sprache« definiert.

```
;*** Sprache für Testprojekt festlegen.
; 16-Bit-Wert für Kombination von
; Computertyp Flag64_128 und Sprache
.Deutsch      = $0110
.Englisch     = $0220
.Sprache      = Deutsch

.COMP64       = $1000
.COMP128      = $2000
.Flag64_128   = COMP64
```

Legt man außerdem noch den Computertyp (hier »:Flag64_128«) fest, dann lässt sich neben einem Infotext in Deutsch oder Englisch auch noch der Computertyp in den Infotext integrieren. Dazu muss man die Konstanten für »:Sprache« und »:Flag64_128« über eine ODER-Verknüpfung miteinander verbinden bzw. auswerten.

Das Ergebnis könnte dann in etwa so aussehen:

```

if .p
    t "TopSym"
    t "TopMac"
    t "LANG.ext"           ;Enthält Sprachvorgabe
endif
    n "TESTPROG"
    o $010e
    z $80

; Infotext für Infoblock definieren.
if Flag64_128 ! Sprache = COMP64 ! Deutsch
    h "Testprogramm"
    h "Nur für GEOS64..."
endif

if Flag64_128 ! Sprache = COMP64 ! Englisch
    h "Test application"
    h "For GEOS64 only..."
endif

if Flag64_128 ! Sprache = COMP128 ! Deutsch
    h "Testprogramm"
    h "Nur für GEOS128..."
endif

if Flag64_128 ! Sprache = COMP128 ! Englisch
    h "Test application"
    h "For GEOS128 only..."
endif

```

Damit erspart man sich bei einem assemblierten Programm den Infotext manuell in den Infoblock der GEOS-Datei schreiben zu müssen.

r Adresse

Über den Pseudo-Opcode *r* kann man eine max. erlaubte Endadresse eines Programms festlegen. Adresse kann dabei als Zahl oder als Symbol definiert werden.

Überschreitet die Programmgröße während der Assemblierung die vordefinierte Grenze, dann wird ein Fehler ausgegeben.



Bild 1.6: Speicherüberlauf bei Verwendung des r-Opcode

In diesem Beispiel soll ein Druckertreiber assembliert werden. Druckertreiber dürfen nach der BSW-Speicheraufteilung den Bereich von \$7900 bis \$7f3f verwenden. Auf Grund eines Fehlers in GeoCalc sollten Druckertreiber aber nur den Speicher bis \$7f3e nutzen, da GeoCalc sonst abstürzen kann.

Im Quelltext wurde zu Beginn die max. erlaubte Endadresse mit dem Opcode *r* auf den Wert \$793e festgelegt. Der Druckertreiber erreicht beim Assemblieren in Pass 1 die Adresse \$7f43 und überschreitet damit den erlaubten Bereich. MA4 gibt in diesem Fall dann eine Fehlermeldung aus.

Wenn man sich die Speicherbelegung für GEOS im **Teil B, Kapitel 1 ab Seite 175** durchließt, dann kennt man die Bereiche die ein GEOS-Programme nutzen darf. Der Opcode *r* scheint daher nicht wirklich notwendig zu sein.

Schreibt man aber eigene Programme, ggf. sogar als VLIR-Projekt, welches dann Code in Form von Modulen nachlädt, dann können sich hier für das Programm evtl. andere Grenzen ergeben, die ein Modul nicht überschreiten darf. Hier kann der Opcode als Sicherheitsüberprüfung eingesetzt werden.

k

Fügt das aktuelle Datum in der (Kurz)Form DDMMYY als Textstring in den Objektcode ein.

Der String wird nicht durch ein *NULL*-Byte beendet, damit kann man nach dem Datum noch weitere Zeichen (z.B. eine Versionsnummer) mit abschließendem *NULL*-Byte anhängen und über *PutString* ausgeben.

l

Fügt das aktuelle Datum in der (Lang-)Form DD.MM.YY als Textstring in den Objektcode ein. Auch hier muss ein evtl. *NULL*-Byte manuell angehängt werden.

x

Der Pseudo-Opcode *x* fügt die aktuelle Uhrzeit in der (Kurz)Form HHMM als Textstring in den Objektcode ein. Wie bei *k* und *l* wird auch dieser String nicht automatisch mit einem *NULL*-Byte abgeschlossen.

y

Fügt die aktuelle Uhrzeit in der (Lang-)Form HH:MM als Textstring in den Objektcode ein. Auch hier muss ein evtl. *NULL*-Byte manuell angehängt werden.

Die Opcodes *k*, *l*, *x* und *y* kann man als automatische Versionsangabe oder Build-Information innerhalb des Programms verwenden. Die Verwendung der Opcodes setzt voraus, das die Systemzeit unter GEOS korrekt eingestellt ist.

g Adresse

Über den Pseudo-Opcode *g* kann man ebenfalls das einhalten einer Bereichsgrenze überprüfen. Im Gegensatz zum Opcode *r* kann der Opcode aber innerhalb eines Programms (auch Mehrfach) und außerhalb eines Programms eingesetzt werden.

Ein Beispiel dafür wäre ein Programm, welches den Hintergrundbildschirm nutzt. Die max. Endadresse wäre daher *BACK_SCR_BASE* (\$6000). Am Ende des Programms wird aber noch ein Zwischenspeicher von ca. 4Kb (= \$1000 Byte) benötigt. Die max. Endadresse für das Programm selbst wäre demnach bei \$5000.

Man kann hier zwar auch den Pseudo-Opcode *r* verwenden, das Problem ist aber das die Größe des Zwischenspeichers sich ggf. noch ändert und dieser hinter dem eigentlichen Programm abgelegt werden soll. Dann würde das Programm beim assemblieren die max. Endadresse von \$5000 einhalten, wenn der Zwischenspeicher aber größer als 4Kb wird, dann würde der Hintergrundbildschirm überschrieben.

Das Beispiel für den Opcode *g* könnte dann wie folgt aussehen:

```
if .p
    t "TopSym"
    t "TopMac"
endif

    n "TEST"
    f APPLICATION
    o $4000

:MAIN    ...                ; Das eigentliche Programm-Dateien

        jmp EnterDeskTop

:tempData    b NULL                ; Letztes Byte bei $5080
:sizeData    = $1000                ; Größe Zwischenspeicher

        g BACK_SCR_BASE - sizeData
```

In dem Beispiel endet das Programm bei \$5080. Die Größe des Zwischenspeichers beträgt \$1000 Byte. Der Opcode *g* testet auf die Bereichsgrenze bei:

BACK_SCR_BASE - sizeData (\$1000 Byte) = \$5000

MA4 gibt dann einen Fehler aus, das der Programmcode bei \$5080 endet und die Bereichsgrenze bei \$5000 überschritten wurde.



Bild 1.7: Speicherüberlauf bei Verwendung des *g*-Opcode

Wenn man sich vor der Programmentwicklung eine eigene Speicherübersicht schreibt und den Speicher strukturiert aufteilt, dann ist auch dieser Opcode eigentlich überflüssig. Aber auch hier kann der Opcode als Sicherheitsprüfung dienen und dabei helfen nicht aus versehen Speicherbereiche zu überschreiben.

e Adresse

Auch der Pseudo-Opcode **e** testet auf Bereichsgrenzen, wird aber in der Regel nur innerhalb eines Programms eingesetzt, da er den Speicher ab der aktuellen Position bis zur angegebenen Adresse mit *NULL*-Byte auffüllt.

Man kann den Opcode zum Beispiel dazu verwenden, wenn bestimmter Programmcode nach dem Opcode immer an einer festen Adresse beginnen soll, der Bereich davon aber noch eine veränderliche Länge hat.

Ein Beispiel dafür wäre ein Variablenspeicher zu Beginn eines Programms und eine definierte Sprungtabelle, die immer an einer festen Adresse beginnen soll.

```
if .p
    t "TopSym"
    t "TopMac"
endif

    n "TEST"
    f APPLICATION
    o APP_RAM
    p JMPTBL

; Programm beginnt bei APP_RAM ($0400)
:VARDATA    b VAR1                ; Programmvariablen
            b VAR2
            ...
            e VARDATA + 256        ; Variablenspeicher max. 256 Bytes

; Sprungtabelle immer ab $0500
:JMPTBL     jmp Adr1              ; Sprungtabelle 1.Eintrag
            jmp Adr2              ; Sprungtabelle 2.Eintrag
            ...
```

Das Label »:JMPTBL« beginnt immer bei \$0500, da der Bereich ab »:VARDATA« bei APP_RAM (\$0400) beginnt und 256 Byte umfassen darf.

Der Pseudo-Opcode **e** füllt den Bereich bis »:JMPTBL« mit *NULL*-Byte auf, bis die Adresse VARDATA + 256 = \$0500 erreicht wurde.

Würde man bei »:VARDATA« mehr als 256 Byte ablegen, dann würde MA4 auch hier die Assemblierung mit einer Fehlermeldung wie beim Pseudo-Opcode **g** abbrechen, da die Bereichsgrenze bei \$0500 überschritten wurde.

q Adresse

Dieser Opcode existiert bereits seit MegaAssembler V2, allerdings wurde bisher die Adresse nicht überprüft: Ist die Adresse kleiner als die Endadresse des Programms, dann wird bei einem DeskAccessory der Ladevorgang durch den GEOS-Kernal abgebrochen (Fehlermeldung "Buffer overflow").

Ab Version V5.2 wurde eine Abfrage ergänzt, die ggf. eine Warnung anzeigt wenn die Adresse des 'q'-Opcode die Endadresse des Programms übersteigt.

Die einzig sinnvolle Verwendung für den 'q'-Opcode ist das Programm "größer" zu machen als es eigentlich ist. Das kann für DeskAccessories hilfreich sein um mehr Speicher im SwapFile auszulagern.

1.3 Der MegaLinker

Neben dem MegaAssembler wurde auch V-Link überarbeitet, genauer gesagt wurde V-Link in MegaAssembler integriert und kann jetzt über das Menü »Verlassen« aufgerufen werden. Aber auch die Menüs wurden etwas optimiert.

Das Menü »Texte«

Wie schon beim MegaAssembler wurde auch im MegaLinker das Menü »Texte« um die Einträge **>>Weiter** und **<<Anfang** erweitert, wenn mehr als 13 Linktexte auf dem eingestellten Laufwerk vorgefunden werden.

Das Menü »Parameter«

Auch hier gibt es neue Parameter die einem das Linken erleichtern sollen:

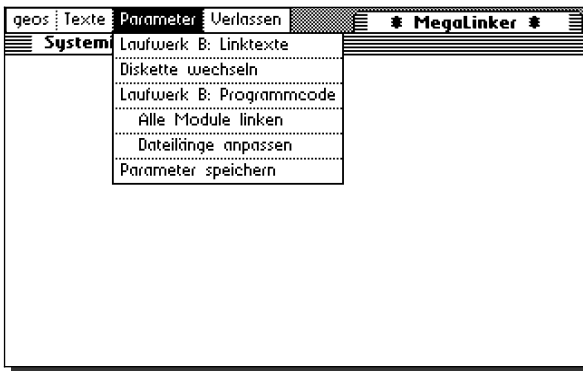


Bild 1.8: Der MegaLinker

Über den ersten Parameter kann man das Laufwerk mit den Linktexten auswählen. Diese müssen nicht zwangsläufig auf dem Programmcode-Laufwerk liegen, das man jetzt getrennt zu den Linktexten auswählen kann. Das Laufwerk für die Linktexte entspricht dem Laufwerk im MegaAssembler für die Quelltexte und das Laufwerk für den Programmcode ist das gleiche wie beim MegaAssembler.

Wenn man die Option »Alle Module linken« aktiviert, dann versucht der Linker alle Module im Linktext zu linkern. Fehlt eine der aufgeführten Dateien, dann erscheint eine Fehlermeldung, ansonsten überspringt der Linker die Datei.

Das abschalten der Funktion macht immer dann Sinn, wenn man nur einzelne Module der VLIR-Anwendung neu assemblieren und linkern will. Assembliert man hingegen immer alle Module neu, dann kann man mit der Option sicherstellen, dass in der VLIR-Anwendung auch alle Module enthalten sind. Sofern Platzhalter im Linktext verwendet werden, dann sollte die Option deaktiviert werden, da sonst eine Fehlermeldung angezeigt wird.

Der Parameter »Dateilänge anpassen« korrigiert die Dateilänge, wenn ein einzelnes VLIR-Modul neu in die VLIR-Anwendung integriert wird. Ansonsten berechnet der **MA4** die Dateigröße an Hand der neu gelinkten Module. Letzteres ist auch das Verhalten wie man es vom **MA2** kennt. Da dann aber die Gesamtgröße der Datei nicht mehr korrekt angezeigt wird, empfiehlt es sich diese Option einzuschalten.

Über den Eintrag »Parameter speichern« lassen sich diese Voreinstellungen dauerhaft im Programm und auf Diskette speichern.

Das Menü »Verlassen«

Das Menü kennt man bereits vom V-Link von **MA2**, im MegaLinker sind lediglich zwei Menüpunkte dazugekommen.



Bild 1.9: Der »Verlassen«-Menü des MegaLinker

Neu ist das man wieder zum MegaAssembler zurückwechseln kann und jetzt ggf. auch zum BASIC des C64 wechseln kann.

Der MegaLinker unterstützt, wie bisher bereits V-Link, die Pseudo-Opcodes **a** (Autor), **n** (Programmname), **c** (GEOS-Klasse, jetzt auch "Vw.xyz") und **i** (Datei-Icon).

Für die Definition der Linkliste wird nach wie vor zu Beginn der Pseudo-Opcode **m** und zum Abschluss das **/** Zeichen benötigt. Die genaue Funktionsweise können Sie im **Teil C, Anhang C ab Seite 357** nachlesen.

Neu ist lediglich der Pseudo-Opcode **h**, der wie beim **MA4** es ermöglicht, einen Infotext für den Infoblock vorzugeben.

1.4 Call MegaAss3

Auch das Programm Call MegaAss wurde auf die Version 3.0 aktualisiert. Damit kann man über das »geos«-Menü von GeoWrite den **MA4** starten. Da es den V-Link als eigenes Programm nicht mehr gibt, wurde das entsprechende Icon entfernt.

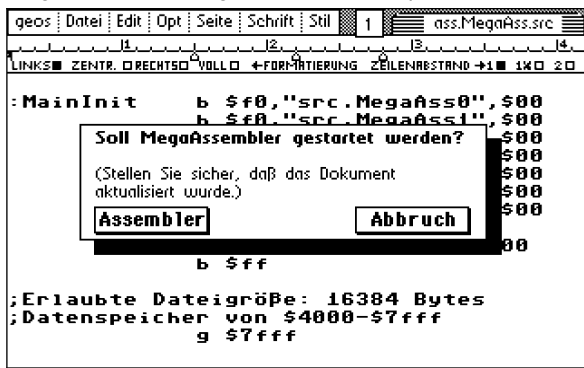


Bild 1.10: Call MegaAss3

1.5 Der AutoAssembler

Was ist der AutoAssembler?

Als diese Funktion in **MA4** eingebaut wurde, entstanden gerade sehr große Projekte. Die Quelltext-Dateien wurden damals auf mehrere 1581-Partitionen auf einer CMD-RAMLink verteilt und mussten nacheinander assembliert werden.

Es war also erforderlich die Assemblierung zwischendurch zu unterbrechen, die Partitionen zu wechseln, um dann die Assemblierung wieder fortzusetzen.

Ohne eine Beschleunigerkarte, wie z.B. die CMD-SuperCPU, musste man sehr viel Zeit vor dem Computer verbringen und jede einzelne Quelltextdatei manuell zum assemblieren auswählen. Dabei entstand die Idee, den gesamten Vorgang zu automatisieren: Die Idee des AutoAssembler war geboren!

Der AutoAssembler benötigt, ähnlich dem Linker, eine Steuerdatei, welche die zu assemblierenden Quelltexte als Dateinamen enthält. Um das System zu vereinfachen sollte das aber keine Textdatei werden, denn die Auswertung der GeoWrite-SteuerCodes wäre zu aufwändig gewesen.

Stattdessen wurde ein binäres Dateiformat gewählt, welches die AutoAssembler-Befehle, die Namen der Quelltexte und ggf. ausführbaren Programmcode enthält.

Programmcode in AutoAssembler-Dateien kann verschiedene Aufgaben während dem assemblieren übernehmen, z.B. kann der Anwender zum wechseln der Quelltext-Diskette aufgefordert werden, es kann auf ein anderes Quelltext-Laufwerk gewechselt werden oder es können temporäre Dateien automatisch gelöscht werden. Auch ist es damit möglich Partitionen auf CMD-Hardware zu wechseln.

Letzteres war damals besonders bei der CMD-RAMLink etwas komplizierter, da GEOS selbst keine Befehle zur Verfügung stellt um die aktive Partition auf CMD-Hardware zu wechseln. Das funktionierte bei der CMD-RAMLink unter GEOS V2.x nur über das setzen von *ramBase* auf die Startadresse der Partition im Speicher.

Auf den folgenden Seiten werden auch Beispiele gezeigt, wie man ggf. Partitionen während dem Assemblierungsvorgang auf CMD-Hardware wechselt. Wir werden dazu aber die erweiterten Befehle aus der Erweiterung "GEOS/MegaPatch" verwenden, da diese wesentlich einfacher zu verwenden sind.

Der AutoAssembler kann aber auch mit mehreren Disketten arbeiten und kann bei Bedarf zum Disk-Wechsel auffordern. Auch diese Funktion muss dann in die AutoAssembler-Befehlsdatei integriert werden.

1.5.1 Die AutoAssembler-Befehle

Es folgt zuerst eine Befehlsübersicht für AutoAssembler-Dateien:

Quelltext wählen \$f0, "Dateiname",0

Nach einem Byte mit dem Wert **\$f0** folgt direkt ein Dateiname, der mit einem NULL-Byte enden muss. Der Dateiname gibt dem AutoAssembler vor, welche Quelltextdatei als nächstes zu assemblieren ist. Die Datei muss sich auf dem aktuell eingestellten Laufwerk für Quelltexte befinden.

Programm ausführen **\$f1**

Der Befehl **\$f1** zeigt dem AutoAssembler an, dass im Anschluss ausführbarer Programmcode (z.B. eine Benutzer-Routine) beginnt. Mit der Hilfe von Benutzer-Routinen kann man dem Anwender mitteilen, das jetzt einer andere Diskette in das Laufwerk gelegt werden muss, das Quelltext-Laufwerk wechseln oder die Partition auf einem CMD-Laufwerk zu wechseln.

Wichtig dabei ist, das man am Ende der Routine im Register **a0** einen Zeiger auf den nächsten AutoAssembler-Befehl hinter dem Programmcode übergibt. Beispiel:

```

        b $f0,"DATEINAME",NULL ; 1.Quelltext

        b $f1                    ; Benutzer-Routine ausführen
        ...                      ; Programmcode

        lda    #<NEXT            ; Zeiger auf nächsten Befehl
        sta    a0L
        lda    #>NEXT
        sta    a0H

        rts                      ; Ende mit RTS

:NEXT    b $f0,"DATEINAME",NULL ; 2.Quelltext
        ...                      ; Weitere Befehle...
```

Innerhalb der Benutzer-Routine ist es auch möglich auf das Quelltext- und Objektcode-Laufwerk zuzugreifen. Dazu wird im Register **a1L** die Adresse für das aktuelle Quelltext-Laufwerk und in **a1H** die Adresse für das festgelegte Objektcode-Laufwerk übergeben.

Im Register **a2L** findet man zusätzlich das Laufwerk, das im Menü »Parameter« für das Quelltext-Laufwerk angegeben wurde.

Laufwerk wechseln **\$f2, Device**

Nach dem Befehl **\$f2** folgt ein Byte, das die neue Adresse für das Quelltext-Laufwerk angibt. Die Adresse darf nur zwischen 8 und 11 (für die GEOS-Laufwerke A: bis D:) liegen.

Zusätzlich kann man den Wert \$00 angeben, dabei wird dann das Laufwerk aktiviert, das im Menü »Parameter« für Quelltexte zugewiesen wurde. Beispiel:

```

        b $f0,"DATEINAME",NULL ; 1.Quelltext

        b $f2,10                ; Laufwerk C: (#10) aktivieren
        b $f0,"DATEINAME",NULL ; 2.Quelltext

        b $f2,11                ; Laufwerk D: (#11) aktivieren
        b $f0,"DATEINAME",NULL ; 3.Quelltext

        b $f2,0                 ; Zurück zum Standard-Laufwerk

        b $f0,"DATEINAME",NULL ; 4.Quelltext
```

Damit kann man z.B. die Quelltexte auf 2xRAM1581-Laufwerke verteilen, und hat damit dann ca. 1600 KByte für Quelltexte zur Verfügung, ohne das man Disketten oder CMD-Partitionen wechseln muss.

MegaAssembler \$f4

Wechselt vom Linker zurück zum MegaAssembler. Anschließend können über den Befehl **\$f0** weitere Quelltexte assembliert werden.

MegaLinker \$f5

Wechselt vom MegaAssembler zum MegaLinker. Anschließend kann über den Befehl **\$f0** ein Dateiname angegeben werden, welcher dann vom MegaLinker zum linken einer VLIR-Anwendung verwendet wird.

Damit ist es möglich mehrere Quelltexte zu assemblieren und anschließend automatisch linken zu lassen, ohne das man als Benutzer vor dem Computer sitzen muss um die einzelnen Dateien auszuwählen. Tritt während der Assemblierung kein Fehler auf, dann bekommt man am Ende ein fertiges VLIR-Programm übergeben!

Endekennung \$ff

Dieser Befehl beendet der AutoAssembler-Modus. Abhängig vom aktuellen Modus kehrt das Programm entweder zur MegaAssembler-Oberfläche oder zum MegaLinker zurück.

Mit Hilfe des AutoAssembler kann der Vorgang bei größeren Projekten durchaus mehrere Stunden dauern, daher wurde eine Abbruch-Funktion integriert.

Während der AutoAssembler Quelltexte bearbeitet und Daten aus der Quelltext-Datei einliest, wird das CIA-Register \$dc01 auf Veränderungen geprüft (z.B. nach einem Tastendruck oder Mausklick) und dann der Vorgang ggf. abgebrochen.

Dieses Verhalten kann im MegaAssembler im Menü »Parameter | Optionen« abgeschaltet werden, siehe »Abbruch(funktion) deaktivieren«.

Um die genaue Funktionsweise der AutoAssembler-Befehle für eine AutoAssembler-Datei zu verstehen, folgen im Anschluss ein paar Beispiele aus der Praxis.

1.5.2 Beispiel für eine AutoAssembler-Datei

Das erste Beispiel ist eine einfache AutoAssembler-Datei, um mehrere Quelltexte hintereinander zu assemblieren und anschließend automatisch aus den einzelnen Modulen eine VLIR-Application zu erstellen.

Dazu benötigen wir als erstes eine GeoWrite-Datei, die als Quelltext zum Erstellen der AutoAssembler-Datei verwendet wird.

Die Quelltext-Datei für den AutoAssembler

Hier das **Listing_D.1**:

```

; --- GEOS-Header
    n "ass.MegaAss"
    c "ass.SysFile V1.0"
    h "Steuerdatei für AutoAssembler"
    f $04 ;GEOS-Filtyp "SYSTEM"

    o $4000

:MainInit    b $f0,"src.MegaAss0", $00
              b $f0,"src.MegaAss1", $00
              b $f0,"src.MegaAss2", $00
              b $f0,"src.MegaAss3", $00
              b $f0,"src.MegaAss4", $00
              b $f0,"src.MegaAss5", $00
              b $f0,"src.MegaAss6", $00

              b $f5                      ; Zum MegaLinker wechseln

              b $f0,"lnk.MegaAss", $00

              b $ff                      ; Ende AutoAssembler-Datei

;Erlaubte Dateigröße: 16384 Bytes
;Datenspeicher von $4000-$7fff
    g $6000                      ; Für das Beispiel ausreichend

```

Wir verzichten bei diesen kurzen Programmen auf die Einbindung von "TopSym" und "TopMac", da die Anzahl der Konstanten überschaubar ist und wir hier auch direkte Zahlenwerte verwenden können.

Für AutoAssembler-Dateien ist im MA4 der Speicherbereich von \$4000-\$7fff (16.384 Bytes) reserviert. Eine AutoAssembler-Datei muss daher zwingend ab \$4000 im Speicher abgelegt werden. Das wird über den Opcode **o** festgelegt.

Die GEOS-Klasse muss grundsätzlich "ass.SysFile V1.0" lauten. Daran erkennt der AutoAssembler die Steuerdateien, die bei aktivierten AutoAssembler-Modus im Menü »Texte« dargestellt werden.

Der Name für die AutoAssembler-Datei kann frei gewählt werden.

Über den Opcode **h** definieren wir hier direkt einen Infotext für den GEOS-Infoblock der später erzeugten AutoAssembler-Datei.

Der GEOS-Filetyp ist \$04 (=SYSTEM).

Als nächstes erwartet der AutoAssembler eine Liste mit Dateinamen für die einzelnen Quelltexte, die später nacheinander assembliert werden sollen. Die Einträge folgen immer dem gleichen Schema:

```
b $f0, "NAME", $00
```

Über den Opcode **b** wird der Eintrag für eine Quelltext-Datei eingeleitet. Es folgt der AutoAssembler-Befehl **\$f0** gefolgt von einem Komma-Zeichen als Feld-Trenner. Danach muss der Dateiname einer Quelltext-Datei in Anführungszeichen folgen. Abgeschlossen wird der Eintrag durch ein \$00-Byte.

Der zur Verfügung stehende Speicher reicht aus um hunderte Dateien aufzulisten, die später automatisch assembliert werden können. In unserem Beispiel sind es nur sieben Quelltexte ("src.MegaAss0" bis "src.MegaAss6").

Als nächstes folgt der Befehl **\$f5**. Wie aus **Teil D, Kapitel 1.5.1 ab Seite 444** bekannt sein dürfte, wird damit zum MegaLinker gewechselt.

Damit der MegaLinker nun weiß, welchen Linktext er verwenden soll um die VLIR-Anwendung zu linken, folgt hier noch der Dateiname der Link-Datei. Dieser wird, wie bei den Quelltext-Dateien auch, durch einen AutoAssembler-Befehl **\$f0** eingeleitet und mit einem \$00-Byte abgeschlossen.

Beendet wird unsere Datei für den AutoAssembler mit dem Befehl **\$ff**. Optional könnte man davor auch mit dem Befehl **\$f4** wieder zum MegaAssembler wechseln.

Am Ende definieren wir mit dem Opcode **g** eine Bereichsgrenze bei \$6000. Würde man z.B. mehr als 450 Dateien auflisten (abhängig von der Länge der Dateinamen), dann würde MA4 beim assemblieren der Steuerdatei einen Fehler melden.

Würde man den gesamten Speicher von \$4000-\$7fff verwenden, dann könnte man rund 900 Dateien einbinden. Das dürfte eher selten der Fall sein. Da man aber auch benutzerdefinierten Programmcode einbinden kann, ist es durchaus sinnvoll die maximale Dateigröße zu testen.

Alternativ zum Opcode **g** könnte man auch den Opcode **r** im GEOS-Header zu Beginn des Quelltextes verwenden. Das könnte dann so aussehen:

```
; --- GEOS-Header
    n "ass.MegaAss"
    c "ass.SysFile V1.0"
    h "Steuerdatei für AutoAssembler"
    f $04 ;GEOS-Filtyp "SYSTEM"

    o $4000
    r $8000 ; Ab $8000 beginnt das GEOS-System!
```

Unsere GeoWrite-Datei ist ein ganz normaler Quelltext, den wir jetzt zuerst in eine binäre AutoAssembler-Datei umwandeln müssen.

Dazu darf im »Parameter«-Menü die Einstellung »AutoAssembler« nicht aktiviert sein (es darf keine Markierung am Anfang der Zeile angezeigt werden!). Anschließend den Quelltext über das »Text«-Menü zum assemblieren auswählen.

Wenn kein Fehler aufgetreten ist, dann haben wir jetzt eine AutoAssembler-Datei, die wir dazu verwenden können, mehrere Quelltexte nacheinander zu assemblieren.

Aus den Systeminformationen kann man ablesen, das unsere AutoAssembler-Datei 113 Byte umfasst (Bereich Objektcode von \$4000-\$4070). Die zuletzt geprüfte Bereichsgrenze war \$6000 (das erlaubte Ende der AutoAssembler-Datei).

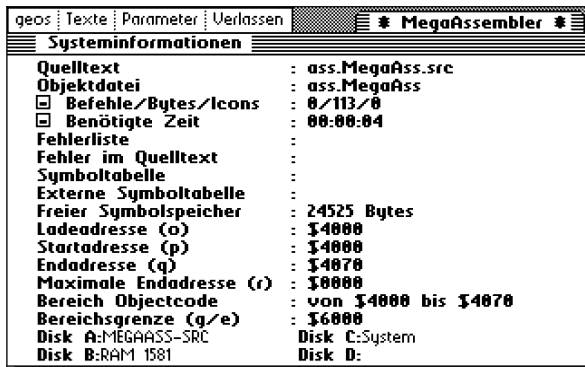


Bild 1.11:
Systeminformationen
des MegaAssembler V4.

Ändern man die Quelltexte für sein Projekt, dann muss die AutoAssembler-Datei nicht angepasst werden. D.h. die einzelnen Arbeitsschritte im vorhergehenden Abschnitt muss man nur einmal ausführen.

Fügt man seinem Projekt allerdings weitere Quelltexte hinzu, oder ändert die Namen der Quelltext-Dateien, dann muss der Quelltext der AutoAssembler-Datei angepasst und neu assembliert werden.

AutoAssembler starten

Als nächstes müssen wir den AutoAssembler-Modus aktivieren. Öffnen Sie dazu das »Parameter«-Menü im **MA4** und aktivieren Sie "AutoAssembler".

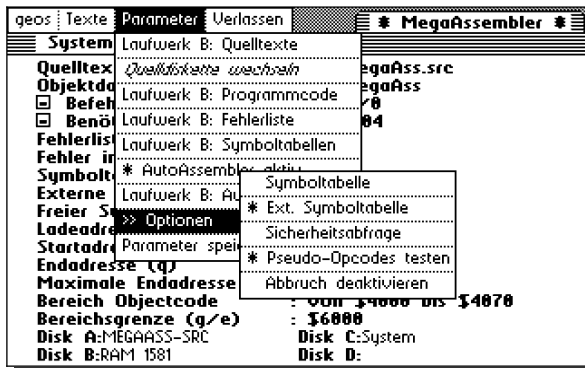


Bild 1.12: Der AutoAssembler
ist aktiviert!

Die anderen Einstellungen müssen Sie passend zu Ihrem Projekt wählen.

In diesem Beispiel sind alle Quelltexte auf dem Laufwerk B, die Ausgabe von Programmcode, Fehlerliste und Symboltabellen soll ebenfalls auf Laufwerk B erfolgen. Die AutoAssembler-Dateien liegen ebenfalls auf Laufwerk B.

Wir benötigen für das Projekt außerdem externe Symboltabellen und lassen zur Sicherheit die Pseudo-Opcodes testen.

Über das Menü »Texte« werden uns jetzt keine Quelltexte mehr angezeigt, sondern nur noch AutoAssembler-Dateien (hier "ass.MegaAss").

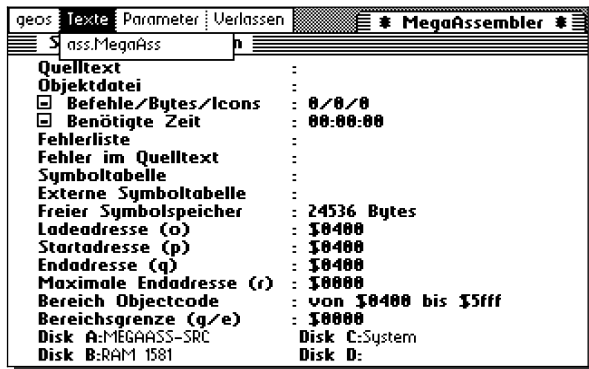


Bild 1.13: Das »Texte«-Menü mit AutoAssembler-Dateien

Auch hier werden, sofern es mehr als 13 AutoAssembler-Dateien gibt, die Menü-Einträge >>Weiter und <<Anfang angezeigt.

Über einen Mausklick auf den entsprechenden Eintrag startet der **MA4** die Arbeit und assembliert der Reihe nach die vorgegebenen Quelltexte und erzeugt aus den assemblierten Modulen über den MegaLinker abschließend die VLIR-Application.

Wenn keine Fehler aufgetreten sind, dann meldet der MegaLinker zum Schluss das der Linkvorgang erfolgreich abgeschlossen wurde.



Bild 1.14: Der Linkvorgang ist beendet

Neben »Text:« findet sich der Name der zuletzt verwendeten Link-Datei. Daneben wird bei »Erzeugte Datei:« noch Name der zuletzt erstellen VLIR-Datei angezeigt.

Damit hat der AutoAssembler alle Aufgaben erledigt und man kann über das Menü »Verlassen« das soeben erzeugte Programm testen, sofern es sich dabei um eine GEOS-Application handelt.

Wir könnten aber auch über das Menü »Verlassen« wieder zur Oberfläche des **MA4** zurückkehren und uns ein paar Informationen zum Projekt betrachten:

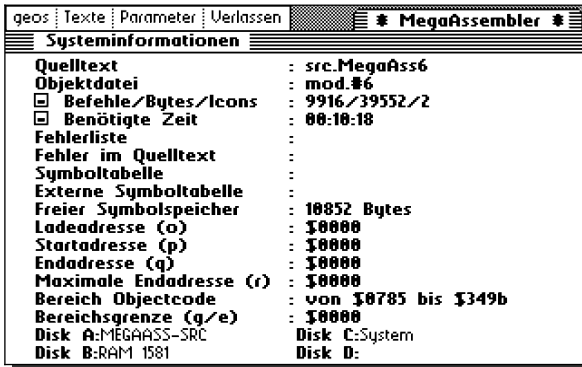


Bild 1.15:
Systeminformationen
zum assemblierten Projekt

Wir sehen das unser Projekt aus 9916 Befehlen besteht, es wurden 39552 Datenbyte verarbeitet und insgesamt finden sich zwei Icons im Projekt, die über den Opcode *j* in den Quelltext eingebunden werden. Zum assemblieren und linken der sieben Quelltexte hat der **MA4** rund 10min. benötigt.

Selbst mit diesem einfachen Beispiel kann erkennen, wie viel Arbeit einem der AutoAssembler-Modus abnehmen kann. Man muss vor allem nicht vor dem Computer sitzen um alle Quelltexte manuell auszuwählen und zu assemblieren. Das alles geht ab jetzt komplett automatisch!

Dadurch das alles automatisch abläuft kann man das assemblieren des MegaAssembler durch den MegaAssembler selbst als eine Art Benchmark betrachten.

Man sollte allerdings sicherstellen das auf dem Ziel-Laufwerk keine Kopie des MegaAssembler existiert, da sonst eine Eingabe des Anwenders erforderlich wird ("Datei ersetzen?").

Ein Richtwert für das automatische assemblieren des MegaAssembler mit einer CMD-SuperCPU und RAM-Disk sind ca. 45Sek.

1.5.3 Programme in AutoAssembler-Dateien

Im nächsten Beispiel sollen Quelltexte von verschiedenen 1541-Disketten assembliert werden. Dazu soll die AutoAssembler-Datei an bestimmten Stellen den Anwender zu einem Disk-Wechsel auffordern.

Dabei können die bereits bekannten GEOS-Routinen verwendet werden, z.B. die Routine *DoDlgBox* um Dialogboxen anzuzeigen. Auch kann man den Bildschirm mit zusätzlichen Hinweisen für den Anwender füllen.

Man sollte dabei aber beachten, das **MA4** beim assemblieren im Bildschirmspeicher der Vordergrundgrafik von \$a000-\$bf3f Programmdatei ablegt. Damit der Anwender beim assemblieren den Programmcode im Bildschirmspeicher ab *SCREEN_BASE* (\$a000) nicht zu sehen bekommt, wird der Speicherbereich ab *COLOR_MATRIX* (\$8c00) mit einem Farbwert gefüllt, der die gleiche Farbe für Vorder- und Hintergrund verwendet.

Wird Programmcode ausgeführt, der auch Ausgaben auf dem Bildschirm erfordert, dann sollte man zuvor den Bildschirminhalt initialisieren. Das schließt auch den Farbspeicher bei *COLOR_MATRIX* ein!

Die folgende AutoAssembler-Datei verwendet dazu ein kleines Unterprogramm, das man dann innerhalb der Steuerdatei beliebig oft aufrufen kann. Es ist also auch möglich, Unterprogramme in AutoAssembler-Dateien einzubinden und diese an verschiedenen Stellen wieder aufzurufen.

Die Quelltext-Datei für den AutoAssembler

Hier zuerst das das **Listing_D.2**, das eine Erweiterung des vorherigen Listing ist.

```
if .p
    t "TopSym"
    t "TopMac"

;--- Diskwechsel-Makro

:GET_DISK    m

    b $f1                ; Benutzer-Routine

    lda    screencolors    ; Bildschirm initialisieren
    jsr    doClrScr

    lda    a2L              ; Quelltext-Laufwerk
    clc                    ; für Dialogbox berechnen
    adc    #"A" -8
    sta    :93

    LoadW   r0,:dlgbox
    jsr     DoDlgBox        ; Dialogbox anzeigen

    lda    a2L
    jsr    SetDevice        ; Quelltext-Laufwerk öffnen

    jsr    OpenDisk         ; Diskette öffnen
```



```

        lda    #$ff                ; Bildschirm wieder löschen
        jsr    doClrScrn

        LoadW a0,:NEXT            ; Weiter mit AutoAssembler
        rts

::dlgbox    b    $81                ; Dialogbox
            b    DBTXTSTR,$10,$0e
            w    :91
            b    DBTXTSTR,$10,$1e
            w    :92
            b    OK,$02,$40
            b    NULL

::91        b    "Bitte die nächste Diskette in",NULL
::92        b    "Laufwerk "
::93        b    "X: einlegen!",NULL

::NEXT
        /

;--- Unterprogramme.
:INIT      m

            b    $f1                ; Benutzer-Routine

        LoadW a0,endClrScrn      ; Weiter mit AutoAssembler
        rts

;--- Bildschirm löschen.
:doClrScrn    sta    :color        ; Bildschirmfarbe speichern

            jsr    i_FillRam        ; Farbspeicher löschen
            w    23*40              ; (Nur Zeilen 2-24)
            w    COLOR_MATRIX +2*40
::color      b    $bf

            lda    #$00            ; Bildschirm löschen
            jsr    SetPattern        ; (Gesamter Bildschirm)
            jsr    i_Rectangle
            b    $00,$c7
            w    $0000,$013f

            rts                    ; Ende Unterprogramm

:endClrScrn
        /
endif

;--- GEOS-Header
        n    "ass.MA1541"
        c    "ass.SysFile V1.0"
        h    "Steuerdatei für AutoAssembler"
        f    SYSTEM
        o    $4000

```

```

;
; AutoAssembler-Code
;
:MainInit      INIT                      ; Unterprogramme einbinden

                b $f0,"src.MegaAss0",$00 ; Quelltexte Teil #1
                b $f0,"src.MegaAss1",$00
                b $f0,"src.MegaAss2",$00
                b $f0,"src.MegaAss3",$00

                GET_DISK                  ; Neue Diskette einlegen

                b $f0,"src.MegaAss4",$00 ; Quelltexte Teil #2
                b $f0,"src.MegaAss5",$00
                b $f0,"src.MegaAss6",$00

                b $f5                      ; Zum MegaLinker wechseln

                b $f0,"lnk.MegaAss",$00  ; Linkdatei

                b $ff                      ; Ende AutoAssembler

;Erlaubte Dateigröße: 16384 Bytes
;Datenspeicher von $4000-$7fff
                g $6000                  ; Für das Beispiel ausreichend

```

Wir haben für das Beispiel die Quelltexte auf zwei 1541-Disketten verteilt. Die Include-Dateien wie "TopSym" oder "TopMac" befinden sich auf einer RAM-Disk: **MA4** sucht Include-Dateien zuerst auf dem Quelltext-Laufwerk und anschließend auf anderen Laufwerken. Das Vorgehen empfiehlt sich für Include-Dateien, die von Dateien auf verschiedenen Disketten benötigt werden. Wenn diese Dateien nicht auf einem permanenten Laufwerk (z.B. das Laufwerk mit dem **MA4**) zur Verfügung stehen, müsste man die Dateien auf alle Quelltext-Disketten kopieren.

Als erstes binden wir die Include-Dateien "TopSym" und "TopMac" mit ein, da wir im Programmteil verschiedene GEOS-Routinen verwenden wollen. Wir definieren hier außerdem noch zwei Makros: »:INIT« und »:GET_DISK«.

Danach folgt der GEOS-Header mit dessen Hilfe wir die Startadresse, GEOS-Klasse, Dateiname, Infotext und den GEOS-Filetyp festlegen.

Im Anschluss an den GEOS-Header beginnt der AutoAssembler-Teil.

Als erstes rufen wir das Makro »:INIT« auf. Das Makro beginnt mit einer Benutzer-Routine **\$f1** und setzt direkt danach den Zeiger auf das Ende des Makros. Der Grund dafür ist das wir hier ein Unterprogramm einbinden, das wir später über **jsr** aufrufen wollen. Da wir die Routine häufiger verwenden werden, benötigt hier das Unterprogramm weniger Speicher als ein weiteres Makro.

Es folgt der erste Teil der Quelltexte. In unserem Beispiel sind es vier Quelltexte ("src.MegaAss0" bis "src.MegaAss3").

Weitere Quelltexte passen nicht auf eine 1541-Diskette. Daher müssen wir nach dem assemblieren von "src.MegaAss3" den Benutzer anweisen, die nächste Diskette in das Quelltext-Laufwerk einzulegen. Dazu binden wir das Makro »:GET_DISK« ein.

Sehen wir uns das Makro »:GET_DISK« etwas genauer an:

```

;--- Diskwechsel-Makro
:GET_DISK      m

                b $f1                ; Benutzer-Routine

                lda  screencolors    ; Bildschirm initialisieren
                jsr  doClrScr

```

Das Makro beginnt wie üblich mit dem Opcode *m*. Es folgt der AutoAssembler-Befehl *\$f1* für eine Benutzer-Routine. Hier rufen wir als ersten »:doClrScr« auf.

Die Routine »:doClrScr« ist ein Unterprogramm, das wir bereits am Anfang über das »:INIT«-Makro in die AutoAssembler-Datei eingebunden haben. Das Unterprogramm initialisiert den Farbspeicher vor der Ausgabe einer Dialogbox, da während dem assemblieren von Quelltexten der Bildschirmspeicher für Programmdaten des **MA4** verwendet wird. Der Farbspeicher wird dabei mit einem speziellen Farbwert gefüllt, der sämtliche Ausgaben auf dem Bildschirm "versteckt".

»:doClrScr« erwartet im Akku eine Bildschirmfarbe. Da wir später eine Dialogbox in dem Bereich anzeigen wollen, verwenden wir die Standard-GEOS-Farben, die im Register *screencolors* gespeichert sind.

Nachdem der Bildschirm gelöscht wurde definieren wir für die Dialogbox das aktuelle Quelltext-Laufwerk, damit wir dem Anwender mitteilen können, in welches Laufwerk er die nächste Disketten einlegen muss.

Anschließend können wir die Dialogbox aufrufen. Dabei ist nichts weiter zu beachten, da die Routine nur einen Text auf dem Bildschirm ausgibt, und anschließend auf das »OK« des Anwenders wartet.

Anschließend aktivieren wir wieder das Quelltext-Laufwerk.

```

                lda  a2L
                jsr  SetDevice        ; Quelltext-Laufwerk öffnen

                jsr  OpenDisk        ; Diskette öffnen

```

In *a2L* übermittelt der MA4 das Laufwerk, welches im Menü »Parameter« für das Quelltext-Laufwerk festgelegt wurde. *SetDevice* könnte ggf. eingespart werden, da wir aber *OpenDisk* aufrufen, um die neue Diskette zu öffnen, sollten wir auch sicherstellen, das wir das richtige Laufwerk aktiviert haben.

Anschließend sollte der Bildschirm wieder initialisiert werden.

```

                lda  #$ff            ; Bildschirm wieder löschen
                jsr  doClrScr

```

Im Gegensatz zum ersten Aufruf von »:doClrScr« übergeben wir im Akku nicht die Bildschirmfarben, sondern den Wert *\$f*.

Der Wert *\$f* steht für Vorder- und Hintergrundfarbe "Hellgrau". Damit wird sichergestellt, das der Inhalt des Bildschirmspeichers "unsichtbar" ist. Wie bereits zuvor erwähnt, verwendet der **MA4** den Bildschirmspeicher für Programmdaten. Würde man hier den Farbspeicher nicht initialisieren, würde man hier während dem assemblieren von Quelltexten nur Pixelmüll zu sehen bekommen.

Nach dem Makro »:GET_DISK« folgen die restlichen Quelltextdateien.

Das war es dann schon fast, wir müssen nur noch zum Linker wechseln und unser Projekt abschließend linken.

```
b $f5                ; Zum MegaLinker wechseln
b $f0, "lnk.MegaAss", $00 ; Linkdatei
b $ff                ; Ende AutoAssembler
```

Wenn wir jetzt die AutoAssembler-Datei auswählen und den AutoAssembler starten, dann beginnt **MA4** mit dem assemblieren der ersten Quelltexte.

Wenn der AutoAssembler zu einem Disk-Wechsel auffordert, dann sieht das in unserem Beispiel wie folgt aus:

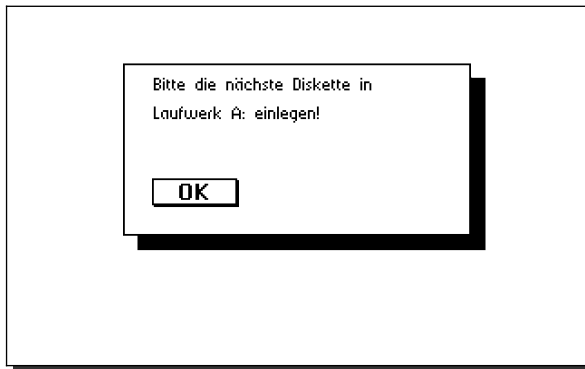


Bild 1.16: Disk wechseln!

Nach einem Mausklick auf »OK« werden die restlichen Quelltexte assembliert. Am Ende wird dann wieder der Linker gestartet und unsere Anwendung wird erstellt.

Man kann das Makro »:GET_DISK« auch dahingehend erweitern, das man dem Anwender mitteilt, welche Diskette einzulegen ist.

Auch ist es ggf. sinnvoll eine Datei auf der neu eingelegten Disk zu suchen, um sicherzustellen, das auch die richtige Diskette im Laufwerk eingelegt ist.

Dazu kann man die Routine *FindFile* verwenden. Evtl. kann es noch sinnvoll sein eine Fehlerbehandlung zu ergänzen.

Auch eine Abbruch-Funktion ist möglich:

Dazu setzt man das Register *a0* vor der Rückkehr zum MegaAssembler auf eine Adresse im Speicher, in welcher der Wert *\$ff* abgelegt ist. Der Wert zeigt dem MegaAssembler an, das hier die AutoAssembler-Datei zu Ende ist und der Vorgang wird abgebrochen.

1.5.4 Weitere Möglichkeiten mit dem AutoAssembler

Wir haben jetzt mit Hilfe der zwei Beispiele gesehen, was mit dem AutoAssembler möglich ist. Bei den Benutzer-Routinen kann man allerdings noch weitere Funktionen integrieren, einige Möglichkeiten sollen hier noch aufgezeigt werden.

Wechseln von Partitionen auf CMD-Laufwerken

Da GEOS V2.x keine Routinen bereitstellt um Partitionen auf CMD-Laufwerken zu wechseln, muss man hier auf Kernalroutinen ausweichen. Eine Alternative wäre der Einsatz von Erweiterungen wie z.B. GEOS/MegaPatch. Da hier nur eine Anregung für weitere Möglichkeiten gegeben werden soll, werden wir uns auf die zusätzlichen Routinen in GEOS/MegaPatch beschränken.

Für dieses Beispiel gehen wir von einer CMD-HD aus. Partition1 enthält den ersten Teil der Quelltext-Dateien, Partition2 den zweiten Teil.

Da wir nur eine Partition wechseln wollen benötigen wir hier keine Dialogbox, sondern nur eine Benutzer-Routine, welche die passende Partition aktiviert. Dafür wurden im folgenden **Listing D.3**: zwei Makros definiert.

```

if .p
    t "TopSym"
    t "TopMac"
    t "TopSym.MP3"           ; Routinen für GEOS/MegaPatch

;--- Partitionswechsel-Makros
:GET_PART1
    m
    b $f1                   ; Benutzer-Routine

    lda    a2L
    jsr    SetDevice        ; Quelltext-Laufwerk öffnen
    lda    #1
    sta    r3H
    jsr    OpenPartition    ; Partition 1 öffnen

    LoadW  a0,:NEXT        ; Weiter mit AutoAssembler
    rts

::NEXT
    /

:GET_PART2
    m
    b $f1                   ; Benutzer-Routine

    lda    a2L
    jsr    SetDevice        ; Quelltext-Laufwerk öffnen
    lda    #2
    sta    r3H
    jsr    OpenPartition    ; Partition 2 öffnen

    LoadW  a0,:NEXT        ; Weiter mit AutoAssembler
    rts

::NEXT
    /
endif

;--- GEOS-Header
    n "ass.MA1541"
    c "ass.SysFile V1.0"
    h "Steuerdatei für AutoAssembler"
    f SYSTEM

    o $4000

```

```

:MainInit      GET_PART1                ; Partition 1 aktivieren

               b $f0,"src.MegaAss0", $00; Quelltexte Teil #1
               b $f0,"src.MegaAss1", $00
               b $f0,"src.MegaAss2", $00
               b $f0,"src.MegaAss3", $00

               GET_PART2                ; Partition 2 aktivieren

               b $f0,"src.MegaAss4", $00; Quelltexte Teil #2
               b $f0,"src.MegaAss5", $00
               b $f0,"src.MegaAss6", $00

               b $f5                    ; Zum MegaLinker wechseln

               b $f0,"lnk.MegaAss", $00 ; Linkdatei

               b $ff                    ; Ende AutoAssembler

;Erlaubte Dateigröße: 8192 Bytes
;Datenspeicher von $4000-$5fff
g $6000

```

Die AutoAssembler-Datei soll zuerst den ersten Teil der Quelltexte assemblieren, dann die Partition wechseln, die restlichen Dateien assemblieren und anschließend die VLIR-Anwendung automatisch linken.

Ab »:MainInit« wird daher zuerst die Partition1 aktiviert, in dem wir das Makro »:GET_PART1« aufrufen. Das Makro ruft *SetDevice* auf, übergibt in *r3H* die Partition1 an *OpenPartition* und setzt im Anschluss die Assemblierung der Dateien fort. *OpenDisk* ist hier nicht erforderlich, da *OpenPartition* das für uns erledigt.

Damit stellen wir sicher, das auf dem Quelltext-Laufwerk auch die richtige Partition aktiviert ist. Ansonsten müsste man vor dem Start des **MA4** auf dem Quelltext-Laufwerk die Partition1 manuell öffnen.

Es folgt der erste Teil der Quelltexte. In unserem Beispiel sind es die vier Quelltexte "src.MegaAss0" bis "src.MegaAss3".

Anschließend rufen wir das Makro »:GET_PART2« auf. Das Makro verwendet *SetDevice* um das Quelltext-Laufwerk zu aktivieren, übergibt in *r3H* die Partition2 an *OpenPartition* und setzt im Anschluss die Assemblierung der Dateien fort.

Anschließend wird der Linker gestartet und die VLIR-Anwendung erstellt.

Wie man erkennen kann, sind die beiden Makros deutlich kürzer als das Makro »:GET_DISK« aus dem vorherigen Listing. Wir müssen hier allerdings auch keinen Benutzerdialog aufrufen und können uns daher das Löschen des Bildschirms sparen.

Löschen von temporären Objektcode-Dateien

Bei größeren Projekten kann es ggf. erforderlich sein, das man Dateien assembliert, die später als Datenfile über den Opcode **d** in eine Anwendung eingebunden werden.

Diese temporären Objektdateien kann man nach dem assemblieren des Projekt löschen, sofern man diese nicht benötigt wenn man ein einzelnes Modul später erneut assemblieren möchte. Gleiches gilt auch für externe Symboldateien.

Auch das löschen solcher Dateien kann man über eine Benutzer-Routine automatisch erledigen lassen. Hier ein Beispiel für eine solche Routine:

```

;--- Temp. Dateien löschen
:CLEANUP      b $f1                ; Benutzer-Routine

                LoadW  r0,:file1    ; Temp. Objektdatei löschen
                jsr    DeleteFile
                LoadW  r0,:file2    ; Ext. Symboldatei löschen
                jsr    DeleteFile
                LoadW  a0,:NEXT      ; Weiter mit AutoAssembler
                rts

::file1       b "obj.TempDatei",NULL
::file2       b "TempDatei.ext",NULL

::NEXT        b $ff                ; Ende AutoAssembler

```

Nach dem Steuercode **\$f1** werden mit Hilfe der Routine *DeleteFile* auf dem aktiven Laufwerk verschiedene Dateien gelöscht. Die Dateinamen müssen alle mit einem *NULL*-Byte beendet werden.

Am Ende muss man die Kontrolle wieder an den MegaAssembler übergeben.

Programm automatisch starten

Eine weitere Möglichkeit für den AutoAssembler wäre das assemblierte Programm am Ende automatisch zu starten.

```

;--- Programm starten
:START        b $f1                ; Benutzer-Routine

                lda     a1H          ; Objektcode-Laufwerk
                jsr     SetDevice
                jsr     OpenDisk      ; Diskette öffnen

                LoadB  r0,%00000000 ; Programm laden und starten
                LoadW  r6,:file      ; Zeiger auf Dateiname

                lda     #>EnterDesktop -1 ; Bei Ladefehler zurück zum
                pha     ; Desktop wechseln
                lda     #<EnterDesktop -1
                pha

                jmp     GetFile       ; Objektfile laden und starten

::file        b "Objektfile",NULL  ; Dateiname Objektfile

```

Da *GetFile* bei einem Ladefehler zur Anwendung zurückkehren würde, legen wir hier die Routine *EnterDesktop* auf dem Stack ab. Dabei ist zu beachten, dass hier die Rücksprungadresse-1 erwartet wird. Damit wird bei einem Ladefehler automatisch der GEOS-Desktop gestartet.

Damit kennen Sie nun die wichtigsten Funktionen des neuen MegaAssembler V4 und des AutoAssembler und können damit Ihre Arbeit ggf. etwas vereinfachen.

1.6 Weitere Änderungen im MegaAssembler

- In Pass#2 wird in der Statuszeile die Programm-Größe angezeigt. Dabei wurde immer die Endadresse +1 angezeigt. Bei einem Programm das von \$e000-\$ffff im Speicher liegt wurde dann hier \$e000-\$0000 angezeigt. Das wurde in **MA4** soweit korrigiert, das die Endadresse = das letzte Byte des Programms darstellt.

Für den Infoblock zeigt die Endadresse aber weiterhin auf "Endadresse +1", da *SaveFile* aus diesen Werten die Anzahl der zu schreibenden Bytes errechnet.

- Wenn der Symbolspeicher überläuft kann es bei MegaAssembler V2 bis V4.2 zu irreführenden Fehlermeldungen oder zu einem Absturz kommen.
- Der Infotext wird aus dem ersten VLIR-Modul in VLIR-Datei übernommen, wenn zuvor noch kein Infotext über den Opcode *h* definiert wurde.
- Fehler beim öffnen der externen Symboltabelle behoben, wenn eine bestimmte Laufwerkskonfiguration verwendet wird (4x RAM-Laufwerk).
- Ab Version V4.9 lässt sich das Abbrechen des Assemblierungsvorgangs über einen Tastendruck unterbinden. Ein versehentliches Abbrechen ist nicht mehr möglich, auch nicht mit der [RUN/STOP]-Taste, vgl. **Teil C, Anhang A ab Seite 327**.
- Ab Version V5.2 wird eine Warnung angezeigt, wenn die durch den 'q'-Opcode definierte Endadresse kleiner als die Endadresse des Programms ist.

1.7 Fehlerkorrekturen

1.7.1 Fehler im MegaAssembler

- In den Routinen "":DefDataFile", "":DefVLIRFile" und "":DefTextFile" wird der Name der angegebenen Datei mit dem Label "":DefFileName" verglichen. Das ist falsch! Richtig wäre der Vergleich mit dem Label "":ObjectFileName" = "Name der Zieldatei". Der angegebene Name darf mit diesem Dateinamen nicht(!) übereinstimmen.
- In der Routine "":IsMakroErrInTab" wird mit falschen Werten verglichen. Wenn innerhalb eines Makros ein Fehler mehrfach auftritt, wird dieser Fehler auch mehrfach in die Fehlertabelle aufgenommen. Stattdessen wird jetzt der Fehler mit "":CurLineData2" verglichen. Damit wird das Makro mit dem Fehler dann nur noch einfach in die Fehler-Tabelle aufgenommen.
- Beim Label "":MakroOpenFlags" Stack mit geöffneten Makros mit 10 Byte zu groß definiert, da maximal 5 Verschachtelungen möglich sind.
- In der Routine "":ClrMakOpenFlags" werden ebenfalls 10 Byte gelöscht, es sind aber nur 5 Verschachtelungen möglich.
- Bei der Routine »:BackToTextFile« wird nach dem Label »:102« die Adresse »:usedRecords« angesprochen. Es findet sich aber kein weiterer Zugriff auf diese Adresse, der Befehl kann also entfallen.
- In der Routine »:FindAssOpcode« wird die Adresse »:Poi_CurLine« doppelt genutzt. In diesem Quelltext durch »:Vec_OpcodeTab« ersetzt. Allerdings nutzen beide noch den gleichen Speicherbereich.

- Die Routinen »:InsLabel2CharArea« und »:DefMakroStartPar« fügen ein Label bzw. Makro in den Symbolspeicher ein. Für den Fall das der Symbolspeicher überläuft wird am Ende des Assemblierungsvorgang versucht die Fehlerliste zu erstellen.
- Da beim MegaAssembler V3 bis V4.2 der Fehlerspeicher durch die Anzeige "Symboltabelle voll" überschrieben wird, stürzt MegaAssembler ab oder erzeugt irreführende Fehler in der Fehlerliste, da die Liste durch die Dialogbox überschrieben wurde. Dieser Fehler wurde in V4.3 behoben, es wird in diesem Fall keine Fehlerliste erstellt.
- Bis zur V4.4 wurden die Operanden von Pseudo-Opcodes nicht auf Gültigkeit getestet. Wird z.B. "f" (GEOS-Dateityp) mit einem nicht definierten Label verwendet, dann wird der Operand mit "0" angenommen. Das erzeugt dann eine Nicht-GEOS-Datei, was bei VLIR-Dateien im Linker zu einem Folgefehler führt (siehe Fehler im Linker). Es gibt daher jetzt im Menü »Parameter« eine Option um eine Überprüfung der Operanden hinter dem Opcode vorzunehmen.
- Die Versionen V2 bis V4.6 erzeugen einen falschen Branch-Befehl, wenn das Sprungziel genau 128 Bytes vorwärts liegt. Dies führt im erzeugten Programmcode zu einem Sprung um 128 Bytes nach hinten, ohne das ein Fehler angezeigt wird.

1.7.2 Fehler im Linker

- In der Routine »:DefFirstGWbyte« muss die erste Zeile **lda #\$19** lauten:
Bei GeoWrite V1.x beginnt die Seite mit 10 Word (Randeinstellungen) plus 4 Byte (NEWCARDSET) plus 2 Byte für die Sektorverkettung. Macht zusammen 26 Byte. Dieser Wert minus eins muss im Assembler-Befehl angegeben werden.
- Fehler beim linken von mehr als 16 Modulen behoben:
Wenn das 17, 18, 19... Modul nicht auf Diskette ist und im Linktext nicht durch ein Semikolon ausgeklammert wurde, erscheint im Linker die Fehlermeldung "Datei nicht gefunden" und der Link-Vorgang wird abgebrochen.
- Der GEOS-Dateityp (Applikation, Hilfsmittel...) wird jetzt ebenfalls aus dem ersten Modul entnommen. Damit können jetzt auch VLIR-Dateien vom Typ "SYSTEM" oder "Hilfsmittel" erzeugt werden.
- V-Link wurde um den Opcode **h** ergänzt:
Damit ist es möglich einen Text in den Infoblock der VLIR-Datei zu übertragen. Jeder **h**-Opcode definiert dabei eine eigene Zeile im Infotext, kann also öfters im Linktext angegeben werden. Am Ende jedes **h**-Opcode wird dann ein CR-Code eingetragen. Es können inkl. Zeilenumbruch max. 95 Zeichen übergeben werden.
- Komplette Überarbeitung des gesamten Quellcode:
Etliche Routinen wurden aus dem Programmcode entfernt, da diese nicht verwendet wurden. Außerdem wurden einige Befehle und Routinen optimiert. Der Programmcode wurde dabei um etwa 8% (705 Byte) reduziert.
- Rückkehr zum Hauptmenü nach »Parameter speichern«. Ohne die Änderung gab es bisher keine visuelle Rückmeldung das die Funktion ausgeführt wurde.
- Dateigröße anpassen wenn Nicht-GEOS-Dateien im Linker in eine VLIR-Datei eingebunden werden sollen. Wenn kein Infoblock vorhanden ist, dann wurde bisher die Dateigröße falsch berechnet.

1.8 Übersicht über alle Fehlermeldungen

Fehler	Beschreibung
\$01	Label unbekannt
\$02	Befehl/Makro unbekannt
\$03	Adressierungsart mit diesem Befehl unmöglich
\$04	Label doppelt definiert
\$05	Bedingter Sprung (branch) zu weit
\$06	Wert zu groß (>\$ff)
\$07	Makroende (/) außerhalb einer Makrodefinition
\$08	ungültige Label-/Makrobezeichnung
\$09	Labelname als Makro gebraucht
\$0a	Makroname als Label gebraucht
\$0b	.) fehlt
\$0c	.(. fehlt
\$0d	Argument fehlt
\$0e	.o. darf nur einmal benutzt werden
\$0f	.if. darf nicht geschachtelt werden
\$10	.else. ohne .if.
\$11	.endif. ohne .if.
\$12	Makros können nicht lokal definiert werden
\$13	Makros können nicht in eine Symboltabelle eingetragen werden
\$14	lokale Labels können nicht in eine Symboltabelle eingetragen werden
\$15	Label ist länger als 63 Zeichen
\$16	Branch-Sprungziel muss ein Label enthalten
\$17	Ungültige Zahlenangabe
\$18	Wert zu groß (>\$ffff)
\$19	Kein Label angegeben
\$1a	Grafik als File-Icon ungeeignet
\$1b	Texte im w-Befehl nicht möglich.
\$1c	String nicht abgeschlossen
\$1d	Überlauf
\$1e	ungültige Makroparameterangabe
\$1f	Anzahl der Makroparameter ungültig
\$20	Fehlender oder bereits vergebener Filename
\$21	max. Makroschachtelungstiefe überschritten
\$22	max. Makroparameteranzahl überschritten
\$23	VLIR-Datensatz nicht ansprechbar
\$24	Grafik als Objektcode-Icon ungeeignet
\$25	ungültiger Filename
\$26	Keinen Textnamen angegeben

Hinweis:

Ist Bit #7 gesetzt, dann trat der Fehler innerhalb eines Makros auf.

1.9 Eintrag in der Labeltabelle

	Info1	Info2	Info3	Info4	Info5
Standard	Byte-Länge	Labelname	Adresse(Word)		
Extern	Byte-Länge + %01000000	Labelname	Adresse(Word)		
Makro	Byte-Länge + %10000000	Makroname	Track	Sektor	BytePos
Lokal	Byte-Länge	:Labelname	Adresse(Word)	\$00 für Label oder \$01 für MakroLabel	

KAPITEL 2

GEOS/MegaPatch

GEOS V1.0 wurde von Berkeley Softworks um 1985 entwickelt, es gab aber nach der Veröffentlichung von GEOS V2.0 im Jahr 1988 keine wirklichen Aktualisierungen mehr.

Es gab zwar in Deutschland noch das deutsche GEOS V2.5, das war aber ein GEOS V2.0 mit ein paar aktualisierten bzw. neuen Programmen, wie etwa der TopDesk. Erweiterungen, z.B. für neue Hardware, mussten dem Betriebssystem auch weiterhin über Patches beigebracht werden.

In den 1990er-Jahren wurde dann mit der Arbeit an einem allgemeinen Patch für GEOS gearbeitet, welcher das GEOS-System mit vielen Fehlerkorrekturen und Verbesserungen ausstatten sollte.

Zu Beginn des Jahres 2000 wurde dann "GEOS/MegaPatch V3" veröffentlicht. Die Erweiterung wird im allgemeinen als "GEOS/MP3", oder noch einfacher, als "MP3" bezeichnet. Seit der ersten Version wurden allerdings einige Fehler gefunden, die erst knapp 20 Jahre später behoben wurden. Zeitgleich wurde dann auch Unterstützung für neue Hardware ergänzt.

Auf den folgenden Seiten wird auf die erweiterten GEOS-Routinen und -Register eingegangen, mit deren Hilfe man neue Hardware nutzen und GEOS-Anwendungen um neue Funktionen erweitern kann.

2.0 Speicherbelegung unter GEOS/MegaPatch

Die Speicheraufteilung unter MP3 entspricht weitestgehend der von GEOS, siehe **Teil B Kapitel 1 ab Seite 175**. Zusätzlich sind folgende Bereiche definiert:

		Beschreibung
\$4000	- \$5fff	Menü für TaskManager
\$4000	- \$55ff	Menü für Druckerspooler
\$6400	- \$7fff	Bildschirmschoner-Routine
\$6d00	- \$78ff	Speicherbereich für das Register-Menü

MP3 setzt zwingend eine Speichererweiterung voraus. Das kann eine Commodore REU, CMD-RAMLink oder CMD-SuperCPU (jeweils mit RAMCard) oder eine GeoRAM bzw. BBGRAM sein (im folgenden weiterhin allgemein als REU bezeichnet).

Wie bei *sysRAMFig* beschrieben, ist Bank 0 für das GEOS-System reserviert. Unter MP3 ist eine REU folgendermaßen belegt:

Bank	Adresse	Beschreibung
0	(Immer Bank #0)	GEOS (RBoot, MoveData, Laufwerkstreiber...)
...		Speicher für Laufwerkstreiber, Programme usw.
X -2	MP3_64K_DISK	Enthält die Laufwerkstreiber für den GEOS.Editor
X -1	MP3_64K_SYSTEM	Vorletzte Speicherbank für Zwischenspeicher von MP3
X	MP3_64K_DATA	Letzte Speicherbank für die erweiterten Routinen von MP3

In den Systemadressen *MP3_64K_SYSTEM* (\$9fa9) und *MP3_64K_DATA* (\$9faa) ist hinterlegt, welche Speicherbank für den Bereich reserviert ist. In *MP3_64K_DISK* (\$9fab) findet sich ggf. optional die Speicherbank mit den Laufwerkstreibern.

Unter MP3 werden Speichererweiterungen bis 16Mb unterstützt, davon werden aber nur die ersten 4Mb für GEOS verwendet. Der restliche Speicher oberhalb von 4Mb kann als RAM-Laufwerk verwendet werden.

Sind mehrere Speichererweiterungen im System vorhanden, dann wird eine der Erweiterungen als Systemspeicher verwendet, die anderen können als zusätzliches RAM-Laufwerk genutzt werden.

2.1 Der Speicherbereich von \$8000-\$8fff

In diesem Bereich wurden nur einige der Systemadressen neu definiert, die restlichen Adressen entsprechen der Beschreibung in **Teil B, Kapitel 1.7 ab Seite 183**.

driveType = \$848e, 4 Byte

Der Bereich ab *driveType* umfasst weiterhin vier Byte (\$848e bis \$8491). Unter MP3 findet man hier für die Laufwerke A, B, C und D jetzt den Emulationsmodus für das entsprechende Laufwerk. Folgende Modi sind definiert:

Wert	Format	Beschreibung
\$00		Kein Laufwerk
\$01	1541	C=1541, CMD-FD/HD-1541, SD2IEC-1541
\$02	1571	C=1571, CMD-FD/HD-1571, SD2IEC-1571
\$03	1581	C=1581, CMD-FD/HD-1581, SD2IEC-1581
\$04	Native	CMD-FD/HD-Native, SD2IEC-Native
\$05	PCDOS	DOS-1581, DOS-FD
\$41	1541S	C=1541 Shadow-Laufwerk
\$81	RAM41	RAM1541, CMD-RAMLink1541
\$82	RAM71	RAM1571, CMD-RAMLink1571
\$83	RAM81	RAM1581, CMD-RAMLink1581
\$84	RAMNM	RAMNative, CMD-RAMLinkNative

Mit Hilfe von *driveType* wird hier nicht mehr der Laufwerkstyp definiert, sondern das Emulations- oder Diskettenformat.

Der Grund für die Neudefinition war die Inkompatibilität anderer GEOS-Desktopoberflächen: Diese definieren für CMD-Geräte neue Formate, so dass z.B. die CMD-RAMLink von anderen Programmen nicht mehr als RAM-Laufwerk erkannt wird. Um auch den echten Laufwerkstyp bestimmen zu können, wurde die Adresse *RealDrvType* eingeführt, die auf den nächsten Seiten beschrieben wird.

ramBase = \$88c7, 4 Byte

Um kompatibel zum GEOS V2.x zu bleiben wurde dieses Register nicht verändert. Auch unter MP3 findet man ab *ramBase* vier Byte (\$88c7 bis \$88ca), welche die Startadresse des aktuellen RAM-Laufwerks in der Speichererweiterung enthalten.

Eine Sonderstellung nimmt hier die CMD-RAMLink ein: In GEOS V2.x findet man in *ramBase+X* das Highbyte der aktiven Partition und in *driveData+3* das zugehörige Lowbyte. Die aktive Partition wird jetzt von den RAMLink-Treibern intern verwaltet, diese sind daher nicht mehr auf diese Adressen angewiesen.

Um aber auch weiterhin den Partitionswechsel über *ramBase* zu erlauben, prüft der RAMLink-Treiber das Highbyte auf Veränderungen.

Findet der Laufwerkstreiber hier einen neuen Wert (z.B. nachdem das Programm "CMD_Move" verwendet wurde), dann wird die zugehörige Partition auf der CMD-RAMLink gesucht und geöffnet.

Das Register *driveData+3* wird nicht mehr benötigt, wird aber aus Kompatibilitätsgründen durch den Laufwerkstreiber gesetzt.

Unter GEOS V2 speichert der Laufwerkstreiber für eine 1571 in *driveData* den Status der aktuellen Diskette (\$00=einseitig, \$80=doppelseitig). Unter MP3 findet man diese Angabe in *doubleSideFlg*.



Hinweis: Die folgenden 16 Byte werden erst seit GEOS V2 durch den GEOS-Kernal beim Start initialisiert und sind laut MemoryMap des C64 in der VIC-Bank bzw. im Bildschirmspeicher ungenutzt (\$xxe8 bis \$xxf7).

sysApplData = \$8fe8, 16 Byte

Der Bereich steht für den DeskTop zur Verfügung, um vor dem Start bzw. nach dem Beenden einer Application bestimmte Einstellungen beizubehalten.

Aufteilung von sysApplData (\$8fe8 bis \$8ff7) unter DeskTop V2

PADCOLDATA = \$8fe8, 8 Byte

Der Bereich wird ab GEOS V2 vom DeskTop dazu genutzt, die Farben für den Arbeitsplatz und Datei-Icons über das Programm "pad color mgr" zu speichern.

Dabei werden für die GEOS-Dateitypen 0-15 in den 16 Nibble der 8 Byte die Vordergrundfarbe für die Icons gespeichert. Es findet keine Bereichsüberprüfung statt, d.h. unbekannte Dateitypen erhalten undefinierte Farben.

Adresse	Typ	Low-Nibble	Typ	High-Nibble
\$8fe8	\$00	Nicht-GEOS	\$01	BASIC
\$8fe9	\$02	Assembler	\$03	Datenfile
\$8fea	\$0e	Systemdatei	\$05	DeskAccessory
\$8feb	\$06	Application	\$07	Dokument
\$8fec	\$08	Zeichensatz	\$09	Druckertreiber
\$8fed	\$0a	Eingabetreiber	\$0b	Laufwerkstreiber
\$8fee	\$0c	Startprogramm	\$0d	Temporär
\$8fef	\$0e	Selbstaussührend	\$0f	Eingabetreiber 128

GEOS/MegaPatch selbst initialisiert die Adressen beim Systemstart über *FirstNinit* mit \$00, greift später aber selbst nicht mehr auf diese Adressen zurück.

DESKPADCOL = \$8ff0

Hier wird von GEOS V2 und DESKTOP V2 die Farbe für den Arbeitsplatz abgelegt.

Wie üblich findet sich im High-Nibble die Farbe für den Vordergrund und im Low-Nibble die Farbe für den Hintergrund.

Auch diese Adresse wird von GEOS/MegaPatch durch *FirstNinit* lediglich initialisiert, MP3 greift aber selbst nicht mehr auf diese Adresse zurück.

unused = \$8ff1, 7 Byte

Der Bereich wird von GEOS/MegaPatch nicht genutzt, lediglich initialisiert.

2.2 Der Speicherbereich von \$9000-\$9d7f

Es folgt eine Beschreibung für die neuen erweiterten Register innerhalb der MP3-Laufwerkstreiber. Diese können nur unter GEOS/MP3 verwendet werden!

DiskDrvTypeExt = \$9074, 4 Byte

Ab dieser Adresse findet sich die folgende Textkennung:

```
:DiskDrvTypeExt      b "DDX", NULL
```

Flag_SD2IEC = \$9078

Diese Adresse wurde in GEOS/MegaPatch V3.3r4 eingeführt und in V3.3r6 verschoben, um innerhalb des Laufwerkstreibers den Wert für *RealDrvMode* zu setzen. Der Wert wird nur innerhalb der 1541/71/81-Laufwerkstreiber und des SD2IEC-Treibers verwendet, ist aber aus Gründen der Kompatibilität in allen Laufwerkstreibern enthalten und kennzeichnet ein SD2IEC-Laufwerk.

Das Flag wird durch die Installationsroutine des Laufwerkstreibers ermittelt und bei der Installation direkt im Laufwerkstreiber gespeichert. Da der Wert über eine ODER-Verknüpfung mit *RealDrvMode* verknüpft wird, enthält *Flag_SD2IEC* entweder den Wert %0000 0010 für "SD2IEC" oder %0000 0000 für "Kein SD2IEC".

Hinweis: Der Zugriff auf diese Adresse sollte nur durch Laufwerkstreiber erfolgen!

Bei allen anderen Laufwerkstreibern findet man hier den Wert \$00.

GeoRAMSize = \$9079

Diese Adresse ist ab GEOS/MegaPatch V3.3r6 in allen Laufwerkstreibern zu finden, wird aber nur im GeoRAM-Native-Treiber verwendet und definiert die aktuelle Bankgröße der GeoRAM-Speichererweiterung. Der Wert wird benötigt um die korrekten Speicheradressen für den Zugriff auf den erweiterten GeoRAM-Speicher zu ermitteln. Mögliche Werte sind:

GeoRAM-Größe	Wert	Bank-Größe
bis 4 MByte	\$10	16 KByte
8 MByte	\$20	32 KByte
16 MByte	\$40	64 KByte

Der Wert wird bei der Installation des Laufwerks ermittelt und hier abgelegt.

Hinweis: Interne Adresse, kann sich in künftigen Versionen ändern. Wird der Wert verändert, dann führt dies zu fehlerhaften Zugriffen auf die GeoRAM!

Bei allen anderen Laufwerkstreibern findet man hier den Wert \$00.

DDRV_EXT_DATA1 = \$907a

Diese Adresse ist ab GEOS/MegaPatch V3.3r6 in allen Laufwerkstreibern zu finden und ist für künftige Anwendungen reserviert. Die Adressen werden innerhalb von MP3 nicht verwendet oder verändert.

Anwendungsprogramme können hier eigene Daten ablegen, z.B. zusätzliche Informationen über das verwendete Gerät.

Siehe hierzu auch die neuen DDX-Funktionen *InitForDDrvOp* und *DoneWithDDrvOp*, um die Daten im Register dauerhaft im Laufwerkstreiber zu speichern.

Hinweis: Die beiden Adressen befinden sich innerhalb des Laufwerkstreibers!

Wenn die Werte durch die Anwendung nicht permanent im Laufwerkstreiber gespeichert werden, dann wird der Wert auf den Standard-Wert (\$00 = Wert nicht initialisiert) zurückgesetzt, wenn der Laufwerkstreiber oder das Laufwerk gewechselt wird. Daher empfiehlt es sich den Wert \$00 zu vermeiden.

Wenn der Wert permanent im Laufwerkstreiber innerhalb des laufenden GEOS-System gespeichert werden soll, dann müssen die Routinen *InitForDDrvOp* und *DoneWithDDrvOp* auf den folgenden Seiten verwendet werden.

Das Wechseln des Laufwerkstreibers (z.B. von HD81 auf HDNM) wird die Adresse immer auf den gespeicherten Wert zurücksetzen. Ansonsten wird MP3 an keiner Stelle diese Werte interpretieren oder verändern.

Diese Adressen sind ausschließlich für Anwendungen zu deren Laufzeit reserviert, bzw. sofern der GEOS.Editor nicht verwendet wird, auch zwischen verschiedenen Anwendungen hinweg.

Hinweis: Bei der Verwendung dieser Adressen über verschiedenen Anwendung hinweg gibt es keine Garantie das die Werte unverändert bleiben. Die Adressen sind ähnlich den Adressen *r0* bis *r15* und können nach der Rückkehr zu einer Anwendung jederzeit undefinierte Werte beinhalten.

DDRV_EXT_DATA2 = \$907b

Wie *DDRV_EXT_DATA1*. Damit stehen zwei Byte für Anwendungen zur Verfügung.

DDrvNMData = \$9082, 8 Byte

Im Bereich ab *DDrvNMData*, speichern die NativeMode-Laufwerkstreiber Angaben zum aktuellen Laufwerk/Speichermedium.

Bis zur Version GEOS/MegaPatch V3.3r6 waren die Adressen nicht in allen Laufwerktreibern vorhanden bzw. lagen an unterschiedlichen Stellen im Treiber. Ab der Version V3.3r6 liegen die Werte jetzt einheitlich im Anschluss an die erweiterten DDX-Register, was sich aber künftig wieder ändern kann.

Hinweis: Der Zugriff auf diese Adressen sollte nur durch Laufwerkstreiber erfolgen!

DiskSize_Lb = \$9082, 1 Byte

DiskSize_Hb = \$9083, 1 Byte

Hier legt die Routine *OpenDisk* die Größe der aktuellen NativeMode-Partition oder Native-RAM-Laufwerk in Kb im Low-/Highbyte-Format ab. Der Wert wird von *CalcBlksFree* verwendet um die Anzahl freier Blocks auf dem Laufwerk zu ermitteln. Dazu wird dieser Wert mit dem Wert 4 multipliziert und in *r3* abgelegt.

Hinweis: Interne Adresse, kann sich in künftigen Versionen ändern.

LastTrOnDsk = \$9084, 1 Byte

Hier wird durch *OpenDisk* die letzte verfügbare Spur auf der aktuellen NativeMode-Partition bzw. RAM-Laufwerk abgelegt. Dieser Wert findet sich im BAM-Sektor \$01/\$02 ab Byte \$08. Da dieser Wert an verschiedenen Stellen benötigt wird, speichert der Laufwerkstreiber den Wert hier ab um nicht jedes mal den zweiten BAM-Sektor von Diskette einlesen zu müssen.

Das bedeutet auch das bei einem Partitionswechsel über die Routine *SwapPartition* mit unterschiedlicher Partitionsgröße zwingend *OpenDisk* aufzurufen ist um diesen Wert zu aktualisieren. *OpenPartition* erledigt dies automatisch.

Hinweis: Interne Adresse, kann sich in künftigen Versionen ändern.

DirHead_Tr = \$9085, 1 Byte

DirHead_Se = \$9086, 1 Byte

Hier wird von der Routine *OpenRootDir* der erste BAM-Sektor \$01/\$01 (ROOT) bzw. von *OpenSubDir* die Adresse des ersten Verzeichnis-Sektors (SUBDIR) abgelegt.

Die Adresse wird unter anderem von der internen Routine *SwapDskNamData* verwendet. Dabei wird beim lesen/schreiben des ersten Block geprüft, ob der Diskettenname ab Byte \$90 eingeblendet werden muss oder nicht. Dies erfolgt bei NativeMode und beim 1581-Format automatisch um kompatibel zum Format einer 1541/1571-Diskette zu bleiben.

Hinweis: Interne Adresse, kann sich in künftigen Versionen ändern.

LastSearchTr = \$9087, 1 Byte

Diese Adresse wird von *SetNextFree* verwendet.

Die Routine sucht zuerst ab der Spur, die in *r3L* übergeben wird, bis zur Spur die in *LastTrOnDsk* abgelegt ist nach einem freien Sektor. Dabei wird *LastSearchTr* auf den Wert von *LastTrOnDsk* gesetzt.

Wird hier kein Block gefunden, dann wird *LastSearchTr* auf den Wert von *r3L* (Beginn der ersten Suche) gesetzt und die Suche am Anfang der Disk fortgesetzt.

Hinweis: Interne Adresse, kann sich in künftigen Versionen ändern.

CurSek_BAM = \$9088, 1 Byte

Hier wird die Sektor-Adresse des BAM-Sektors in *dir3Head* abgespeichert. Der Bereich dient hier als Cache um zu vermeiden das der gleiche BAM-Sektor mehrmals gelesen bzw. geschrieben wird.

Hinweis: Interne Adresse, kann sich in künftigen Versionen ändern.

BAM_Modified = \$9089, 1 Byte

Wenn im aktuellen BAM-Sektor in *dir3Head* ein Block belegt oder freigegeben wird, dann wird dieses Flag gesetzt. Damit wird vor dem lesen des nächsten BAM-Sektors nach *dir3Head* sichergestellt, das der aktuelle BAM-Sektor auf Disk geschrieben wird, wenn dieser zuvor verändert wurde.

Hinweis: Interne Adresse, kann sich in künftigen Versionen ändern.

2.3 Der Speicherbereich von \$9d80-\$9fff

In diesem Bereich befinden sich unter MP3 einige neue Systemadressen. Diese sind sowohl unter GEOS/MegaPatch64 als auch unter GEOS/MegaPatch128 verfügbar.

DskDrvBaseL = \$9f7e, 4 Byte

Der Bereich ab *DskDrvBaseL* umfasst vier Byte (\$9f7e bis \$9f81). Hier findet sich das Lowbyte der Adresse, aber der die Laufwerkstreiber für Laufwerk A bis D im erweiterten Speicher der REU, Bank#0, abgelegt werden.

Eine Übersicht zur Speicherbelegung von Bank#0 in der REU findet sich im **Teil D, Anhang K ab Seite 540**.

DskDrvBaseH = \$9f82, 4 Byte

In *DskDrvBaseH* (\$9f82 bis \$9f85) findet man das zu *DskDrvBaseL* passende Highbyte der Adresse.

doubleSideFlg = \$9f86, 4 Byte

Der Inhalt von *doubleSideFlg* (\$9f86 bis \$9f89) enthält bei einem 1571-Laufwerk die Information "einseitig" (\$00) oder "doppelseitig" (\$80).

Diese Werte wurden früher in *driveData* abgelegt, kollidierten aber mit den Angaben im Laufwerkstreiber für die CMD-RAMLink, der in *driveData*+3 das Lowbyte der Startadresse der aktiven Partition abgelegt hat.

drivePartData = \$9f8a, 4 Byte

In *drivePartData* (\$9f8a bis \$9f8d) findet man nach dem Aufruf von *OpenDisk* die aktive Partition auf CMD-Laufwerken (CMD-HD/FD/RAMLink).

RealDrvType = \$9f8e, 4 Byte

Der Bereich ab *RealDrvType* (\$9f8e bis \$9f91) gibt Auskunft über das angeschlossene Laufwerk (\$00 = Kein Laufwerk).

Die unteren 3 Bit (b0 bis b2) definieren auch hier das Emulationsformat, die oberen 5 Bit (b3 bis b7) definieren den tatsächlichen Laufwerktyp:

Wert	Binär	Beschreibung
\$01	%0000 0001	C=1541 / SD2IEC-1541
\$02	%0000 0010	C=1571 / SD2IEC-1571
\$03	%0000 0011	C=1581 / SD2IEC-1581
\$04	%0000 0100	SD2IEC-Native / IECBUS-Native (erst seit 2018)
\$05	%0000 0101	C=1581 - DOS-Modus
\$11	%0001 0001	CMD-FD2000/4000, 1541-Partition
\$12	%0001 0010	CMD-FD2000/4000, 1571-Partition
\$13	%0001 0011	CMD-FD2000/4000, 1581-Partition
\$14	%0001 0100	CMD-FD2000/4000, NativeMode-Partition
\$15	%0001 0101	CMD-FD2000/4000, DOS-Modus
\$21	%0010 0001	CMD-HD, 1541-Partition
\$22	%0010 0010	CMD-HD, 1571-Partition
\$23	%0010 0011	CMD-HD, 1581-Partition
\$24	%0010 0100	CMD-HD, NativeMode-Partition

Wert	Binär	Beschreibung
\$31	%0011 0001	CMD-RAMLink, 1541-Partition
\$32	%0011 0010	CMD-RAMLink, 1571-Partition
\$33	%0011 0011	CMD-RAMLink, 1581-Partition
\$34	%0011 0100	CMD-RAMLink, NativeMode-Partition
\$41	%0100 0001	C=1541 / SD2IEC-1541 mit ShadowRAM (Cache)
\$81	%1000 0001	RAM1541
\$82	%1000 0010	RAM1571
\$83	%1000 0011	RAM1581
\$84	%1000 0100	RAMNative
\$a4	%1010 0100	RAMNative, C=REU-Laufwerk (erst ab 2018)
\$b4	%1011 0100	RAMNative, GeoRAM-Laufwerk (erst ab 2018)
\$c4	%1100 0100	RAMNative, SuperRAM-Laufwerk

Mit Hilfe von *RealDrvType* kann man den aktuellen Laufwerkstyp bestimmen.

Eine Ausnahme stellt das SD2IEC dar, das sich mit den Laufwerken vom Typ 1541/71/81 den Laufwerkstreiber teilt. Es gibt aber dennoch eine Möglichkeit ein Disketten-Laufwerk von einem SD2IEC-Laufwerk zu unterscheiden, nämlich *RealDrvMode*.

Weitere Laufwerkstypen können noch definiert werden. Es sollte jedoch darauf geachtet werden, das kein Laufwerkstyp doppelt belegt wird und das die älteren Programme über bestimmter Bit-Werte in *driveType* den Laufwerkstyp erkennen.

RealDrvMode = \$9f92, 4 Byte

Der Bereich ab *RealDrvMode* (\$9f92 bis \$9f95) gibt Auskunft über die zusätzlichen Eigenschaften für das entsprechende Laufwerk A bis D:

Wert	Binär	Konstante	Eigenschaft
Bit 7	%1000 0000	SET_MODE_PARRTITION	Laufwerk unterstützt Partitionen
Bit 6	%0100 0000	SET_MODE_SUBDIR	Laufwerk unterstützt Native-Verzeichnisse
Bit 5	%0010 0000	SET_MODE_FASTDISK	RAM-Laufwerk oder CMD-HDD/PP-Kabel
Bit 4	%0001 0000	SET_MODE_SRAM	CMD-SuperRAM-Laufwerk
Bit 3	%0000 1000	SET_MODE_CRAM	C=REU-Laufwerk
Bit 2	%0000 0100	SET_MODE_GRAM	GeoRAM-Laufwerk
Bit 1	%0000 0010	SET_MODE_SD2IEC	SD2IEC-Laufwerk
Bit 0	%0000 0001	-	Nicht verwendet

Damit kann die Abfrage des aktuellen Laufwerkstyp entfallen:

Es wird einfach nur das entsprechende Bit in *RealDrvMode* getestet und danach werden die entsprechenden Unterprogramme aufgerufen (z.B. CMD/Partition oder SD2IEC/Disk-Image wechseln).

Die Konstanten *SET_MODE_SRAM*, *SET_MODE_CRAM* und *SET_MODE_GRAM* definieren RAM-Laufwerke, welche außerhalb des erweiterten GEOS-Speichers angelegt werden. Diese Laufwerke nutzen den Bereich oberhalb der ersten 4Mb in der Speichererweiterung, der von GEOS/MegaPatch verwendet und verwaltet wird. Damit können RAM-Laufwerke bis zu einer Größe von 16Mb eingerichtet werden.



Hinweis: Die RAM-Laufwerke für C=REU und GeoRAM wurden erst 2018 in MP3 ergänzt. Je nach verwendeter Hardware können ein C=REU- und GeoRAM-Laufwerk auch gemeinsam installiert werden, z.B. mit der Erweiterung "TurboChameleon64" für den C64.



Hinweis: `SET_MODE_SD2IEC` wurde erst 2019 (ab Version V3.3r4) ergänzt und kennzeichnet ein SD2IEC-Laufwerk mit dem 1541/71/81- oder NativeMode-Laufwerkstreiber. In Verbindung mit `RealDrvType` kann ein Gerät eindeutig als Commodore- oder SD2IEC-Laufwerk erkannt werden.



Hinweis: Frühe Versionen von GEOS/MegaPatch (bis V3.3r3) benötigten für das SD2IEC im 1541/71/81-Modus die "file-based M-R emulation", also ein Laufwerks-ROM, welches über den "XR"-Befehl aktiviert wurde. Aktuell ist das nicht mehr erforderlich.

RamBankInUse = \$9f96, 16 Byte

Hier liegt die zentrale Speicherbelegungstabelle von MP3 für die ersten 4Mb der angeschlossenen Speichererweiterung. Diese Tabelle zeigt an, welche der verfügbaren Speicherbänke frei oder belegt sind.

Jeder Speicherbank sind dabei zwei Bit zugeordnet. Innerhalb eines Byte werden die Bänke von links nach rechts (beginnend mit dem höchsten Bit 7) durchnummeriert:

Byte 0	%11xxxxxx	Bank 0	...	
Byte 0	%xx11xxxx	Bank 1	Byte 14	%11xxxxxx Bank 56
Byte 0	%xxxx11xx	Bank 2	Byte 14	%xx11xxxx Bank 57
Byte 0	%xxxxxx11	Bank 3	Byte 14	%xxxx11xx Bank 58
Byte 1	%11xxxxxx	Bank 4	Byte 14	%xxxxxx11 Bank 59
Byte 1	%xx11xxxx	Bank 5	Byte 15	%11xxxxxx Bank 60
Byte 1	%xxxx11xx	Bank 6	Byte 15	%xx11xxxx Bank 61
Byte 1	%xxxxxx11	Bank 7	Byte 15	%xxxx11xx Bank 62
...			Byte 15	%xxxxxx11 Bank 63

Um nun feststellen zu können wie eine Speicherbank belegt ist, wurden folgende Kombinationen für die Bit-Paare definiert:

Bit-Paar	Belegung
%00	Speicherbank ist nicht belegt
%01	Durch Anwendungen belegt
%10	Für Laufwerkstreiber reserviert
%11	Durch das GEOS-System belegt

Speicher, der durch das GEOS-System belegt ist, kann nur durch die dazugehörige Anwendung (z.B. TaskManager, Spooler usw.) wieder freigegeben werden. Sofern man eigenen Speicher benötigt, sollte man die Bit-Paare testen und nur freie Speicherbänke (Wert %00) verwenden.

Es empfiehlt sich die Speicherbank dann als "Durch Anwendung belegt" zu markieren (%01), da sonst bei einem Wechsel der aktiven Anwendung über den TaskManager der Speicher durch andere Anwendung doppelt belegt werden könnte.

Um eine freie Speicherbank zu suchen, kann man folgende Routine verwenden:

```

:FindFreeBnk    ldy    #$00
::51           jsr    GetBankByte    ; Bank-Status testen.
                beq    :52           ; Bank gefunden, Ende
                iny           ; Nächste Bank.
                cpy    ramExpSize    ; Ende erreicht ?
                bne    :51           ; Nein, weiter...
                ldy    #$00           ; Keine freie Bank...
::52           rts

;*** Bankstatus einlesen.
; Übergabe: yReg=Bank-Adr.
; Rückgabe: AKKU=Status (Bit7+6, %00=frei)
:GetBnkByte    tya           ; Zeiger auf Bankbyte
                lsr           ; berechnen.
                lsr
                tax
                lda    RamBankIsUse,x ; Bankbyte einlesen.
                pha
                tya           ; Zeiger auf Bitpaar
                and    #%00000011    ; isolieren.
                tax
                pla
::51           cpx    #$00           ; Bitpaar in Bit 6+7
                beq    :52           ; verschieben.
                asl
                asl
                dex
                bne    :51
::52           and    #%11000000    ; Bit 6+7 isolieren.
                rts                ; Bankstatus im Akku Bit7/6.

```

Wer die Speicherbank nur kurzfristig benötigt, und es keine Möglichkeit gibt zwischenzeitlich andere Programme ausführen zu lassen, der muss hier nichts weiter beachten.

Wenn allerdings das Programm teilweise beendet wird, zwischenzeitlich die Mainloop abläuft oder der TaskManager gestartet werden kann, dann muss hier die Speicherbank als belegt gekennzeichnet werden (mit %01 für Anwendung).

Man läuft sonst Gefahr das eine andere Anwendung die gleiche Speicherbank als "Frei" erkennt und für eigene Zwecke verwendet.

Maximal stehen hier 16 Bytes für 64 Speicherbänke (0-63) = 4 MByte zur Verfügung. Über *ramExpSize* kann man feststellen ob die Speicherbank überhaupt verfügbar ist.

Bei der CMD-SuperCPU existiert noch die Möglichkeit einen Teil des RAMCard-Speichers zu verwenden. Welche Bereiche frei oder belegt sind, kann man über die entsprechenden RAMCard-Register erfahren. Sofern die RAMCard als GEOS-DACC verwendet wurde, ist der GEOS-Speicher in der SuperCPU als "belegt" markiert.

RamBankFirst = \$9fa6, 1 Word

Definiert die Startadresse des GEOS-Speichers (DACC) in der Speichererweiterung.

Bei einer GeoRAM oder C=REU ist hier immer der Wert \$0000 zu finden. Bei der CMD-RAMLink findet man hier die Startadresse der aktiven DACC-Partition. Bei einer SuperCPU/RAMCard findet man hier die Adresse der erste freien Speicherbank, falls vor dem Start von GEOS bereits Speicher belegt wurde.

Die Adresse wird auch von der Routine *BootGeos* (\$c000) verwendet, um bei einer CMD-RAMLink bzw. CMD-SuperCPU den richtigen Speicherbereich für den schnellen Neustart von GEOS/MP3 zu ermitteln. Wenn der Speicher ab \$9fa6 allerdings überschrieben wurde, dann führt der Aufruf von *BootGeos* zu einem Absturz, da dann die benötigte ReBoot-Routine von GEOS nicht gefunden werden kann. In dem Fall kann "GEOS.RBOOT" verwendet werden, hier wird der Speicherbereich innerhalb des Programms gespeichert.

GEOS_RAM_TYP = \$9fa8

Liefert Informationen zur aktuellen Speichererweiterung unter GEOS. *GEOS_RAM_TYP* kann folgende Werte annehmen:

Wert	Konstante	DACC-Typ
\$10	RAM_SCPU	CMD-SuperCPU / RAMCard
\$20	RAM_BBG	GeoRAM / BBGRAM
\$40	RAM_REU	C=REU
\$80	RAM_RL	CMD-RAMLink / DACC-Partition

Wird von einem Programm eine bestimmte Speichererweiterung angefordert, so kann man folgendes Beispiel dazu verwenden:

```

lda    GEOS_RAM_TYP
cmp    #RAM_SCPU
bne    ERROR

```

In diesem Beispiel wird das Programm beendet, wenn keine CMD-SuperCPU mit RAMCard als GEOS-DACC verwendet wird.

MP3_64K_SYSTEM = \$9fa9

Hier steht die Adresse der 64K-Bank, welche den erweiterten GEOS/MegaPatch-Kernal beinhaltet. Dieses ist in der Regel die letzte verfügbare GEOS-Speicherbank innerhalb des erweiterten GEOS-Speicher.

MP3_64K_DATA = \$9faa

Hier findet man die Adresse der Speicherbank, in der MP3 verschiedene Daten auslagert, wie z.B. das Swapfile oder die Hintergrundgrafik. Die Speicherbank ist in der Regel die vorletzte Speicherbank im erweiterten GEOS-Speicher.

MP3_64K_DISK = \$9fab

In dieser Speicherbank wird von MP3 bei Bedarf eine Kopie der verfügbaren Laufwerkstreiber abgelegt. Wenn diese bereits beim Start von MP3 eingelesen werden (oder nachträglich über die entsprechende Option im GEOS.Editor), dann wird die Datei "GEOS.Disk" nicht mehr benötigt. Findet man hier den Wert \$00, dann werden die Laufwerkstreiber von Diskette eingelesen.

Flag_Optimize = \$9fac

Findet man hier den Wert \$00, so wird die CMD-SuperCPU für GEOS optimiert. Für einige Programme muss die Optimierung jedoch deaktiviert werden (geoBasic und GeoProgrammer/Debugger). Das deaktivieren kann über den GEOS.Editor erledigt werden. Der Wert \$03 schaltet die Optimierung für GEOS aus.

millenium = \$9fad

Hier findet man für 4-stellige Jahreszahlen die Angabe zum "Jahrhundert". Mit der Version von 2018 wurde dieser Wert standardmäßig auf \$14 = 20xx gesetzt.

Flag_LoadPrnt = \$9fae

Dieses Flag kann über den "GEOS64.Editor" geändert werden: Ist der Wert \$00, dann wird der Druckertreiber immer von Diskette gestartet. Hat *Flag_LoadPrnt* den Wert \$80 (Standard), so wird der Druckertreiber aus dem RAM geladen.

Dieses Flag ist nur in GEOS/MegaPatch64 von Bedeutung, da unter GEOS128 der Druckertreiber immer aus dem FrontRAM ab \$d9c0 geladen wird. Daher kann im "GEOS128.Editor" dieses Flag auch nicht geändert werden.

PrntFileNameRAM = \$9faf, 17 Byte

Hier findet man den Namen des im RAM gespeicherten Druckertreibers. Dieser muss nicht mit *PrntFilename* (\$8465) übereinstimmen, da andere Anwendungsprogramme den Treiber wechseln können, indem diese Programme in *PrntFilename* den Name des neuen Druckertreibers eingetragen.

Wenn der Druckertreiber geladen werden soll, dann vergleicht die erweiterte Routine *GetFile* von MP3 den Inhalt von *PrntFilename* mit *PrntFileNameRAM*. Sind beide Bereiche identisch, dann wird der Druckertreiber direkt aus dem RAM eingelesen. Sind die beiden Bereiche unterschiedlich, dann wird der Druckertreiber von Diskette geladen. Danach wird der neue Druckertreiber in den erweiterten Speicher kopiert, von wo aus er beim nächsten Mal direkt eingelesen wird.

Um einen neuen Druckertreiber (auch im erweiterten Speicher) zu installieren, genügt es also auch weiterhin nur den Namen ab *PrntFilename* zu ändern, der Rest wird dann von MP3 beim nächsten Aufruf von *GetFile* automatisch erledigt.

Flag_Spooler = \$9fc0

Ist Bit 7 gesetzt, so ist der Druckerspooler installiert. Die Bit 5 bis Bit 0 dienen als Zähler, der bei jedem Mausklick/Tastendruck zurückgesetzt wird. Ist der Zähler abgelaufen, dann wird Bit 6 gesetzt, und beim nächsten Durchlauf der Mainloop das Spoolermenü gestartet.

Flag_SpoolMinB = \$9fc1

Erste Speicherbank für Druckerspooler. Dieser Wert sollte nur über den GEOS.Editor verändert werden (Registerkarte "Speicher/Spooler").

Flag_SpoolMaxB = \$9fc2

Letzte Speicherbank für Druckerspooler. Zusammen mit *Flag_SpoolMinB* geben diese beiden Register Auskunft über die Größe des für den Druckerspooler reservierten Bereichs in der Speichererweiterung.

Flag_SpoolADDR = \$9fc3, 3 Byte

Zeiger auf die aktuelle Position im Spooler-RAM. An diese Position wird das nächste Byte geschrieben. Format: Lowbyte, Highbyte, Bank-Adresse. Die Bank-Adresse muss zwischen den beiden Werten in *Flag_SpoolMinB* und *Flag_SpoolMaxB* liegen!

Flag_SpoolCount = \$9fc6

Verzögerungszähler für Druckerspooler. Dieser Wert wird mit *Flag_Spooler* verknüpft und kann im GEOS.Editor angepasst werden.

Ist hier Bit 7 gesetzt, so wird das Spoolermenü nicht automatisch gestartet, das Spoolermenü kann dann nur über den TaskManager mit **[CBM]+[CTRL]**, Register "Drucker/Druckerspooler", gestartet werden.

Flag_SplCurDok = \$9fc7

Angabe der aktuellen Dokument-Nummer in der Spooler-Warteschlange.

Flag_SplMaxDok = \$9fc8

Max. Anzahl Dokumente in Spooler-Warteschlange. Es können max. 15 Dokumente in die Warteschlange aufgenommen werden.

Flag_TaskAktiv = \$9fc9

Der Wert \$00 bedeutet, das der TaskManager aktiviert ist. Sollte es erforderlich sein das der TaskManager zeitweise nicht gestartet werden darf, dann kann man hier den Wert \$FF eintragen. MP3 übergeht dann in der Mainloop die Tastaturabfrage zur Aktivierung des TaskManager. Man sollte allerdings den Originalinhalt des Register zwischenspeichern und später wieder zurückschreiben.

Flag_TaskBank = \$9fca

Systemspeicherbank für den TaskManager.

Wenn der TaskManager gestartet werden soll, dann wird das Menü aus dieser Bank mit dem Speicher im C64 getauscht und das Menü gestartet. In dieser Speicherbank wird dazu dann auch der komplette Speicher des C64 gesichert.

Beim beenden des Menüs wird dann der ursprüngliche Speicherinhalt wieder hergestellt und der TaskManager wieder in die Systemspeicherbank kopiert.

Daher darf dieser Wert durch den Anwender auch nicht geändert werden, da sonst der MP3-Kernal den TaskManager nicht mehr findet.

Flag_ExtRAMinUse = \$9fcb

Verschiedene Speicherbereiche in der REU sind für die Dialogbox und Hilfsmittel (DeskAccessories) reserviert. Sind diese Speicherbereiche belegt (Dialogbox geöffnet oder Hilfsmittel gestartet), dann darf der TaskManager den Task nicht wechseln, da sonst innerhalb der anderen Anwendung ein weiteres Hilfsmittel gestartet werden könnte. Damit würde der reservierte Speicherbereich des Swapfile der ersten Anwendung zerstört.

Wenn in *Flag_ExtRamInUse* eines der folgenden Bits gesetzt ist, dann kann der TaskManager die Anwendung nicht wechseln:

```
%1xxx xxxx  $80  Hilfsmittel ist geöffnet
%x1xx xxxx  $40  Dialogbox ist geöffnet
```

Flag_ScrSvCnt = \$9fcc

Aktivierungszeit für Bildschirmschoner. Die genaue Zeit in Sekunden bis zum Start kann im GEOS.Editor eingestellt werden.

Flag_ScrSaver = \$9fcd

Statusbyte für den Bildschirmschoner. Die Bedeutung der Werte im einzelnen:

```
%1xxx xxxx  $80  Bildschirmschoner ist deaktiviert
%x1xx xxxx  $40  Aktivierungszeit zurücksetzen
%xx1x xxxx  $20  Aktivierungszeit herunter zählen
           $00  Bildschirmschoner-Effekt starten
```

Flag_CrsrRepeat = \$9fce

Dieses Register definiert die Zeichen-Wiederholfrequenz der Tastatur. Ein Wert von 3 (\$03) ist optimal, der Wert 15 (\$0f) entspricht der Frequenz unter GEOS V2.x.

BackScrPattern = \$9fcf

Wird im GEOS.Editor die Option "Anzeige/Hintergrund/Hintergrundbild verwenden" deaktiviert, so verwendet der MP3-Kernal das hier angegebene GEOS-Füllmuster zum löschen des Bildschirms.

Flag_SetColor = \$9fd0

Dieses Register definiert wie Farben in Dialogboxen gesetzt werden. Folgende Werte können hier eingesetzt werden:

%1xxx xxxx	\$80	SET_DBOXCOL_ON	Farbe immer setzen
%x1xx xxxx	\$40	SET_DBOXCOL_STD	Farbe nur bei Standard-Dialogbox
	\$00	SET_DBOXCOL_OFF	Dialogbox ohne Farbe anzeigen

Der GEOS.Editor wechselt zwischen den Zuständen \$00 und \$80. Der Wert \$40 ist nur wenig sinnvoll, wurde aber für künftige Erweiterungen integriert.

Flag_ColorDBox = \$9fd1

Dieses Byte wird von der Dialogbox-Routine berechnet. Findet man hier den Wert \$ff, so werden in der aktuellen Dialogbox keine Farben gesetzt.

Diese Funktion bleibt ohne Wirkung wenn es sich um eine Dateiauswahlbox handelt, da diese generell mit Farben gezeichnet wird.

Flag_IconMinX = \$9fd2

MP3 stellt für alle Icons in einer Dialogbox nun auch Farbe zur Verfügung. Allerdings kann es passieren das kleinere Icons nicht im Format von 8x8 Pixel vorliegen.

Deshalb verwendet MP3 erst dann ein Farbe für Icons, wenn diese eine hier definierte Mindestgröße (in Cards) besitzen. Standardmäßig müssen Icons mindestens 5 Cards (40 Pixel) breit sein, damit MP3 die Farbe für das Icon aktiviert. Dieser Wert entspricht der Breite der Icons »OK«, »Abbruch« und »Öffnen«.

Bei der Dateiauswahlbox wird Farbe für alle Icons verwendet.

Flag_IconMinY = \$9fd3

Auch für die Höhe (in Pixel) eines Icons gibt es Mindestgröße, ab der MP3 Farbe verwendet. Standardmäßig findet man hier den Wert \$10 = 16 Pixel. Das entspricht ebenfalls den Angaben der Systemicons.

Flag_IconDown = \$9fd4

Icons können in x-Richtung nur im Bereich einzelner Cards positioniert werden, in y-Richtung gilt diese Einschränkung nicht.

Da die C64 Farbe aber immer nur innerhalb von Cards setzen kann, muss der Kernal die Icons verschieben. Je nach Differenz zum nächsten Card ermittelt MP3 den günstigsten Wert anhand von *Flag_IconDown*.

Standardmäßig findet man hier den Wert \$05, d.h. wenn ein Icon 5 Pixelzeilen unterhalb der letzten Card-Grenze liegt, dann wird es nach unten verschoben.

Flag_DBoxType = \$9fd5

Hier findet man nach dem Aufruf der Dialogbox eine Kopie des Kopfbyte der Dialogboxtabelle. Wird nur intern vom GEOS-Kernal verwendet.

Flag_GetFiles = \$9fd6

Wenn die Routine *DoDlgBox* aufgerufen wird, dann untersucht MP3 zuerst die gesamte Tabelle nach den Steuercodes *DBGETFILES* und *DBUSRFILES*.

Wird einer dieser beiden Steuercodes in der Dialogboxtabelle verwendet, dann wird später die erweiterte Dateiauswahlbox gestartet.

MP3 setzt in *Flag_GetFiles* für *DBGETFILES* das Bit 7 und für *DBUSRFILES* zusätzlich noch das Bit 6 und unterbindet damit alle Bildschirmausgaben des Kernal zum Zeichnen der Dialogbox. Lediglich Icons werden noch bearbeitet, da diese für die Dateiauswahlbox benötigt werden. Die Dateiauswahlbox wird dann erst später durch eine spezielle Kernal-Routine auf dem Bildschirm dargestellt.

DB_GFileType = \$9fd7

Bei einer Dateiauswahlbox mit *DoDlgBox* und *DBGETFILES* muss man in *r7l* den GEOS-Filetyp ablegen, dieser wird vom GEOS-Kernal nach *DB_GFileType* kopiert.

DB_GFileClass = \$9fd8, 1 Word

Hierher kopiert das Kernal den Zeiger auf die GEOS-Klasse aus dem Register *r10* für die Dateiauswahl über *DoDlgBox* mit *DBGETFILES*.

DB_GetFileEntry = \$9fda

Dieses Byte zeigt auf den gewählten Eintrag in der Dialogboxtabelle. Für eine Dialogbox über *DBGETFILES* hat dieser Wert keinerlei Bedeutung, da die Liste der Dateinamen nicht im Speicher abgelegt wird.

Bei einer Auswahlbox über *DBUSRFILES* hat man hier die Möglichkeit auch die Nummer des gewählten Eintrages zu erfahren. Das kann z.B. notwendig sein, wenn man die Liste der Texteinträge intern mit einer zweiten Liste verknüpft hat.

Dies macht sich z.B. der "GEOS.Editor" bei der Laufwerksauswahl zunutze: Hier benötigt der "GEOS.Editor" nicht den Namen des gewählten Laufwerktyps, sondern nur den Wert in *DB_GetFileEntry*. Dieser zeigt dann auf eine interne Tabelle mit den in "GEOS.Disk" verfügbaren Laufwerkstreibern.

DB_StdBoxSize = \$9fdb, 6 Byte

Hier findet man die Werte für die Größe einer Standard-Dialogbox (Dialogboxtabelle mit gesetztem Bit 7 im Kopfbyte der Tabelle):

\$9fdb	Byte	Obere y-Koordinate = \$20
\$9fdc	Byte	Untere y-Koordinate = \$7f
\$9fdd	Word	Linke x-Koordinate = \$0040
\$9fdf	Word	Rechte x-Koordinate = \$00ff

Flag_SetMLine = \$9fe1

Dieses Register definiert die horizontalen und vertikalen Trennlinien innerhalb von PullDown-Menüs. Findet man hier den Wert \$80, so werden Menüs wie unter GEOS V2.x mit Trennlinien zwischen den einzelnen Menüeinträgen dargestellt. Der Wert \$00 zeichnet keinerlei Trennlinien.

Werden keine Trennlinien gezeichnet, dann sollte in *Flag_MenuStatus* das Bit 7 (= Eintrag unter Mauszeiger invertieren) gesetzt werden, damit man bei der Auswahl eines Eintrages innerhalb eines PullDown-Menüs leichter erkennen kann, welcher Eintrag ausgewählt wird wenn sich der Mauszeiger zwischen zwei Einträgen befindet.

Flag_MenuStatus = \$9fe2

Dieses Register definiert das Aussehen von Menüs über *DoMenu*:

Bit 7 ist für das invertieren des aktuellen Menüeintrages verantwortlich. Ist es gesetzt, dann wird der aktuelle Menü-Eintrag beim überfahren mit dem Mauszeiger automatisch invertiert ("Mouse-Over"-Effekt). Wenn der Mauszeiger den Menü-Eintrag wieder verlässt, dann wird der Menü-Eintrag wieder normal angezeigt.

Bit 6 ist für das verlassen des Menüs nach unten zuständig. Ist Bit 6 gesetzt, so kann man Menüs nicht mehr nach unten verlassen. Dies funktioniert allerdings nur wenn der Programmierer dies generell zulässt (durch setzen von *UN_CONSTAINED* im Kopfbyte der Menütabelle). Ist das verlassen von Menüs nur nach oben möglich (*CONSTRAINED*), dann bleibt dieses Bit ohne Funktion.

DM_LastEntry = \$9fe3, 6 Byte

Zeigt auf den Bereich des aktuellen Menüeintrages. Wird intern von der Routine *DoMenu* benötigt um den aktuellen Eintrag zu invertieren.

DM_LastNumEntry = \$9fe9

Zeigt auf den invertierten Eintrag in der Menütabelle.

2.4 Farbtabelle für GEOS/MegaPatch

Die Farbwerte für MP3 liegen ab *MP3_COLOR_DATA* (\$9fea) im Speicher, der Bereich umfasst 22 Byte. Fast alle Farben werden aus der folgenden Tabelle ausgelesen. Wenn einzelne Farben im System angepasst werden sollen, dann kann hier der entsprechende Farbwert geändert werden.

Adresse	Konstante	Funktion
\$9fea	C_Balken	Scrollbalken
\$9feb	C_Register	Aktiver Registerkarten-Reiter
\$9fec	C_RegisterOff	Inaktiver Registerkarten-Reiter
\$9fed	C_RegisterBack	Hintergrundfarbe für Registerkarten
\$9fee	C_Mouse	Mauszeiger
\$9fef	C_DBoxTitel	Dialogbox-Titelzeile
\$9ff0	C_DBoxBack	Dialogbox-Hintergrund
\$9ff1	C_DBoxDIcon	Dialogbox-Icons
\$9ff2	C_FBoxTitel	Dateiauswahlbox-Titelzeile
\$9ff3	C_FBoxBack	Dateiauswahlbox-Hintergrund
\$9ff4	C_FBoxDIcon	Dateiauswahlbox-Icons
\$9ff5	C_FBoxFiles	Dateiauswahlbox-Dateifenster
\$9ff6	C_WinTitel	Fenster-Titelzeile
\$9ff7	C_WinBack	Fenster-Hintergrund
\$9ff8	C_WinShadow	Fenster-Schatten
\$9ff9	C_WinIcon	Fenster-Icons
\$9ffa	C_PullDMenu	PullDown-Menüs Hinweis: GEOS unterstützt von sich aus keine Farbe in PullDown-Menüs. Sollen die Menüs in Farbe angezeigt werden, dann muss das setzen der Farben über eine eigene Routine erfolgen.
\$9ffb	C_InputField	Eingabefelder
\$9ffc	C_InputFieldOff	Inaktives Optionsfeld
\$9ffd	C_GEOS_BACK	GEOS-Hintergrundfarbe
\$9ffe	C_GEOS_FRAME	GEOS-Bildschirmrahmenfarbe
\$9fff	C_GEOS_MOUSE	GEOS-Mauszeiger (Kopie von <i>C_Mouse</i>)



Hinweis:

Das Programm "TopDesk" von GEOS/MegaPatch von 1999/2000 ändert einige der hier aufgeführten Werte um seine eigenen Dialogboxen in der richtigen Farbe darzustellen. Das hat dann auch Auswirkungen auf andere Anwendungen.

Grundsätzlich sollte auf diese Adressen aber nur "lesend" zugegriffen werden oder es sollte dem Anwender gegenüber zumindest klargestellt werden, das hier gemachte Änderungen Systemweit gelten werden.

2.5 Angepasste Kernalroutinen

In MP3 wurden einige der Kernalroutinen von GEOS um neue Funktionen erweitert. Im Folgenden werden nur die Änderungen beschrieben. Die genaue Funktionsweise der Routinen kann im **Teil B, Kapitel 2 ab Seite 204** nachgelesen werden.

Dialogboxroutinen (Ergänzung zu Teil B, Kapitel 3, Seite 215)

3.1 DoDlgBox (\$c256)

Die Definition einer Dialogboxtabelle wurde um einige zusätzliche Funktionen erweitert. Außerdem wurde das Kopfbyte erweitert. Hier eine Übersicht:

Das erweiterte Kopfbyte

Bit 7 aktiviert weiterhin die Standard-Dialogbox. Bit 0 bis Bit 4 definieren auch unter MP3 das Füllmuster für den Schatten der Dialogbox (%00000=kein Schatten).

Bit 6 definiert in MP3 den Farbmodus der Dialogbox: Ist das Bit 6 über *DBOXCOLON* gesetzt, dann wird die Dialogboxtabelle um eine Farbinformation erweitert:

```
:DlgBoxTab      b %00000001 ! DBOXCOLON ; Kopfbyte: Farbe und Schatten
                  b $20,$7f           ; Größe der Dialogbox
                  w $0040,$00ff
                  b Farbe              ; Farbe der Dialogbox
```

Nach den Größenangaben für die Dialogbox muss ein Byte für die Farbe der Dialogbox folgen. Dies funktioniert aber nicht bei Standard-Dialogboxen (Bit 7 muss 0 sein)!

Der Farbwert setzt sich aus der Vordergrundfarbe (High-Nibble, Bit 7 bis Bit 4, 0-15) und der Hintergrundfarbe (Low-Nibble, Bit 3 bis Bit 0, 0-15) zusammen.

Außerdem muss nun bei jedem Icon-Eintrag die Farbe für das Icon mit angegeben werden. Nachfolgend ein Beispiel für eine Icon-Definition:

```
b DBUSRICON          ; Benutzer-Icon
b $01,$10,Farbe      ; X/y-Position, Farbe
w IconEntry

b OK                 ; System-Icon »OK«
b $01,$20,Farbe      ; X/y-Position, Farbe
```

Die Ergänzung durch Farbwerte ist nur dann erforderlich, wenn die Dialogbox immer mit bestimmten Farben am Bildschirm angezeigt werden sollen. MP3 setzt die Farbe für Dialogboxen ohne Bit 6 = *DBOXCOLOFF* mit Hilfe der Systemfarben (siehe **Teil D, Kapitel 2.4 ab Seite 479** - "Farbtabelle von GEOS/MegaPatch").

Wenn Bit 6 gelöscht ist, dann entscheiden die beiden Register *Flag_IconMinX* (\$9fd2) und *Flag_IconMinY* (\$9fd3) ab welcher Größe Icons von MP3 in Dialogboxen automatisch eingefärbt werden sollen. Vorgabe ist 5 Cards breit, 16 Pixel hoch.

Wird Bit 5 gesetzt, dann wird keine Farbe ausgegeben. Eine Dialogbox erscheint dann in den aktuellen Bildschirmfarben, die im Bereich ab *COLOR_MATRIX* liegen.

Die beiden Funktionen Bit 6 (Dialogbox in Farbe) und Bit 5 (Farbe in Dialogbox unterdrücken) schließen sich gegenseitig aus.

Werden beide Bit gesetzt, dann verwendet MP3, unabhängig vom Wert in *Flag_SetColor*, die Farbe wie in der Farbtabelle angegeben. Die Dialogboxtabelle darf dann allerdings keine Farbinformationen beinhalten!

Diese Einstellung ist nur dann sinnvoll wenn ein Hintergrundbild verwendet wird und der Anwender im GEOS.Editor die Option »Dialogboxen in Farbe« deaktiviert hat: Hier würde dann ohne ein gesetztes Bit 6 und Bit 5 die Dialogbox in den Farben des aktuellen Hintergrundbildes dargestellt werden.

Sind jedoch Bit 6 und Bit 5 gesetzt, dann zeichnet MP3 die Dialogbox in jedem Fall mit den in der Farbtabelle definierten Systemfarben.

Funktionsblock

Die Dialogbox versteht in MP3 die folgenden zusätzlichen Dialogbox-Codes:

Code	Konstante	Bedeutung	Parameter
7	DRIVE	System-Icon	Byte, rel. x-Koord. in Cards Byte, rel. y-Koord. in Pixel
8	DUMMY	Füllbyte	Keine Funktion
9	DBUSRFILES	Listenauswahl	Word, Ablagebereich für Listeneinträge
10	DBSETCOL	Farbausgabe	Byte, rel. x-Koord. in Cards Byte, rel. y-Koord. in Cards Byte, Breite in Cards Byte, Höhe in Cards Byte, Farbwert

DRIVE

Dieser Steuercode bindet die vier Laufwerkicons A bis D in die Dialogbox mit ein. Nach *DRIVE* folgt ein Byte für die x-Koordinate in Cards und ein Byte für die y-Koordinate in Pixel. Das Wechseln des Laufwerks übernimmt die Dialogbox allerdings nicht, dazu kann man aber die folgende Routine verwenden:

:doDlgBox	Loadw	r0,dlgBoxData	
	jsr	DoDlgBox	; Dialogbox aufrufen.
	lda	sysDBData	; Laufwerk wechseln?
	bpl	:10	; => Nein, weiter...
	and	##00001111	
	jsr	SetDevice	; Laufwerk aktivieren.
	txa		; Fehler?
	bne	:exit	; => Ja, Abbruch...
	jsr	OpenDevice	; Diskette öffnen.
	txa		; Fehler?
	bne	:exit	; => Ja, Abbruch...
	beq	:doDlgBox	; Dialogbox erneut anzeigen.

Wird ein Laufwerksicon angeklickt, dann findet man in *sysDBData* folgende Werte:

\$88	DBOXDRVA	= Laufwerk A
\$89	DBOXDRVB	= Laufwerk B
\$8a	DBOXDRVC	= Laufwerk C
\$8b	DBOXDRVD	= Laufwerk D

DUMMY

Dieser Befehl wirkt wie der Assembler-Befehl *nop* und hat keine besondere Wirkung. Er kann als Füllbyte innerhalb einer Dialogboxtabelle verwendet werden.

Ein Anwendungsbeispiel wäre z.B. die Original-Dialogboxtabelle in GeoWrite zum erstellen eines neuen Dokuments. Diese könnte bisher wie folgt ausgehen haben:

```
:DlgDefBox      b %10000001
                ...
                b DBUSRICON , $0a, $48
                w ICON_ENTRY_LFWERK
                ...
```

Hier kann man das Icon "LfWerk" durch die vier Laufwerkicons darstellen, wenn der Code für *DBUSRICON* durch *DRIVE* ersetzt wird.

Da für *DRIVE* keinen Zeiger auf einen Iconeintrag benötigt wird, muss hier der Zeiger auf die Icon-Grafik durch zweimal *DUMMY* ersetzt werden. Außerdem muss die x-Koordinate korrigiert werden, da das "LfWerk"-Icon 6 Card breit ist, die vier Laufwerkicons jedoch bis zu 4x2 =8 Cards in Anspruch nehmen können.

Allerdings muss auch der Programmcode nach beenden der Dialogbox angepasst werden, damit an Stelle des nächsten Laufwerks das gewählte Laufwerk aktiviert wird.

Für verschiedene Anwendungen sind Patches verfügbar, welche Dialogboxen in bestehenden Programmen an die neuen Funktionen von MP3 anpassen.

DBUSRFILES

Öffnet eine Auswahlbox mit einer Liste von Einträgen:

```
b DBUSRFILES
w vecFNameTab
```

Das Label »:vecFNameTab« zeigt dabei auf einen Speicherbereich mit bis zu 255 Einträgen. Jeder Eintrag besteht aus 16 Zeichen und einem *NULL*-Byte.

Der gewählte Eintrag aus der Liste wird in *DB_GetFileEntry* übergeben.

Ab Version V3.3r4 lässt sich *DBUSRFILES* mit *DBSETDRVICON* verknüpfen. Damit können die Laufwerkicons A bis D dargestellt werden.

```
b DBUSRFILES ! DBSETDRVICON
w vecFNameTab
```

Mit der Routine *FindFTypes* und dem Dateityp \$ff und *DBUSRFILES* lässt sich so eine kompakte Verzeichnisanzeige realisieren. Kürzer geht es mit *DBGETFILES* und dem Wert \$ff als GEOS-Filetyp in *r7L*.

DBSETCOL

Erstellt ein Rechteck mit frei wählbarer Farbe:

```

                b DBSETCOL
                b xl                ; rel. x-Koordinate Links in Cards
                b yo                ; rel. y-Koordinate Oben in Cards
                b xb                ; Breite in Cards
                b yh                ; Höhe in Cards
:usercol        b col              ; Farbwert

```

Die Koordinaten "xl" und "yo" sind als Bytewerte in Cards anzugeben und bezeichnen die linke, obere Ecke des Rechtecks. "xb" und "yh" geben die Breite bzw. die Höhe des Rechtecks in Cards an.

Für "col" kann ein Farbwert von 0 bis 15 für die Zeichenfarbe (High-Nibble) und für den Hintergrund (Low-Nibble) eingesetzt werden. Die einzelnen Farbwerte entsprechen dabei den C64- Standardfarben.

Sofern die Farbe je nach Aufruf der Dialogbox unterschiedliche Werte verwenden soll, kann man vor Aufruf der Dialogbox den Wert in die Dialogboxtabelle kopieren:

```

LoadB  usercol,$12          ; Weiße Schrift auf rotem Grund
                                ; Benutzerdefinierte Farbe

LoadW  r0,ErrorBox
jsr    DoDlgBox             ; Dialogbox aufrufen

```

DBGETFILES

Übergibt man in *r7L* den Wert 255, dann werden alle Dateitypen angezeigt.

Der Funktionscode *DBGETFILES* wurde um zwei Funktionen erweitert:

DBSELECTPART

Mit dieser Funktion lässt sich eine Partitionsauswahlbox umsetzen.

Setzt man beim Funktionscode *DBGETFILES* das Bit 7, z.B. über eine ODER-Verknüpfung mit der Konstante *DBSELECTPART*, dann öffnet MP3 eine Auswahlbox mit den auf dem Laufwerk verfügbaren Partitionen. Eine Dialogboxtabelle für die Partitionsauswahl sieht dann wie folgt aus:

```

:DlgSlctPart  b %10000001
                b DBGETFILES ! DBSELECTPART, $00, $00
                b OPEN  , $00, $00
                b CANCEL, $00, $00
                b NULL

```

DBSELECTPART ist mit %10000000 definiert. Bei der Dateiauswahlbox werden alle Icons automatisch platziert, weshalb eine Positionsangabe für Icons und die Auswahlbox entfallen kann (X/Y-Koordinaten=\$00).

Wählt man aus der Liste die Partition und bestätigt die Auswahl mit »Öffnen«, dann wechselt MP3 auf dem eingestellten Laufwerk in die ausgewählte Partition. Es muss hier zwingend *OPEN* sein, das System-Icon *OK* beendet lediglich die Auswahlbox!

DBSETDRVICON

Setzt man bei Verwendung von *DBGETFILES* das Bit 6, dann stellt MP3 für jedes verfügbare Laufwerk ein entsprechendes Icon zur Verfügung.

Wenn es möglich sein soll, innerhalb der Auswahlbox auf ein anderes Laufwerk wechseln zu können, dann aktiviert man die Laufwerkicons für die Dateiauswahlbox wie folgt:

```
:DlgSlctFile  b %10000001
               b DBGETFILES ! DBSETDRVICON, $00, $00
               b OK      , $00, $00
               b CANCEL, $00, $00
               b NULL
```

DBSETDRVICON ist mit %01000000 definiert. Innerhalb der Auswahlbox stehen jetzt die Icons »OK«, »ABBRUCH« sowie für jedes verfügbare Laufwerk ein Icon (A, B, C, D) zur Verfügung. Ist eines der Laufwerke innerhalb von GEOS nicht konfiguriert, dann wird das entsprechende Icon für das Laufwerk nicht angezeigt.

Über den Inhalt von *sysDBData* kann man dann erfahren, welches Laufwerk aktiviert werden soll, vgl. Beschreibung zum Funktionscode *DRIVE* auf Seite 481. Dort findet sich auch ein Beispiel wie man das Laufwerk wechselt.

Mit den hier beschriebenen Ergänzungen zur Dateiauswahlbox in MP3 sollte jeder eine passende Möglichkeit finden, um speziell programmierte Dialogboxen in eigenen Anwendungen zu vermeiden.



Bild 2.1: Dateiauswahlbox mit Laufwerkicons

Diskettenroutinen (Ergänzung zu Teil B, Kapitel 12, Seite 282)

12.1.4 FindFTypes (\$c23b)

Diese Routine erstellt eine Tabelle mit Dateinamen des gleichen GEOS-Filetyp. Jeder Dateiname hat eine Länge von 16 Zeichen mit einem abschließenden *NULL*-Byte. In *r7l* erwartet die Routine den GEOS-Filetyp.

Gegenüber früheren GEOS-Versionen kann hier jetzt auch der Wert \$ff angegeben werden. Damit werden dann alle Dateien des aktuellen Verzeichnisses eingelesen. Die Anzahl der Dateien ist grundsätzlich auf 255 Dateien begrenzt.

2.6 Neue Kernalroutinen für GEOS

MP3 stellt auch eine Reihe an neuen Routinen zur Verfügung. Dazu existiert ab \$c0dc im GEOS-Kernal eine erweiterte Sprungtabelle.

Neue Grafikroutinen (Ergänzung zu Teil B, Kapitel 4 ab Seite 223)

4.30 RecColorBox (\$c0e5)

Mit Hilfe der Routine *RecColorBox* lässt sich ein Rechteck mit beliebiger Farbe am Bildschirm zeichnen. Dabei wird nur das FarbRAM in *COLOR_MATRIX* gezeichnet, der Grafikbildschirm wird nicht verändert.

Die Routine erwartet in *r5L* die x-Koordinate, in *r5H* die y-Koordinate, in *r6L* die Breite und in *r6H* die Höhe des Rechtecks. Alle Werte müssen in Cards angegeben werden. Es können also nur Farbflächen in ganzen Cards gezeichnet werden. Zusätzlich wird in *r7L* der Farbwert erwartet. Der Farbwert setzt sich aus der Vordergrundfarbe (High-Nibble, Bit 7 - 4 = 0-15) und der Hintergrundfarbe (Low-Nibble, Bit 3 - 0 = 0-15) zusammen.

Verändert werden der Akku, das x- und y-Register, sowie die Register *r5* bis *r8*.

```
:ClrScrCol      LoadB  r5L,$00          ; rel. x-Koordinate Links in Cards
                  LoadB  r5H,$00          ; rel. y-Koordinate Oben in Cards
                  LoadB  r6L,$28          ; Breite in Cards
                  LoadB  r6H,$19          ; Höhe in Cards
                  LoadB  r7L,$02          ; Farbwert
                  jsr     RecColorBox
```

Verwendet man an Stelle von **LoadB** eine Kopierschleife um die Werte nach *r5* bis *r7L* zu schreiben, dann lassen sich hier ein paar Byte an Programmcode einsparen.

4.31 i_ColorBox (\$c0df)

Die Routine *i_ColorBox* ist die Inline-Routine zu *RecColorBox*. Die Parameter werden aber nicht in den Speicherstellen *r5* bis *r7L* übergeben, sondern werden direkt hinter dem Aufruf übergeben. Für *i_ColorBox* sieht der Aufruf dann so aus:

```
jsr     i_ColorBox
b xl          ; rel. x-Koordinate Links in Cards
b yo          ; rel. y-Koordinate Oben in Cards
b xb          ; Breite in Cards
b yh          ; Höhe in Cards
b col         ; Farbwert
```

Der Aufruf entspricht hier der Funktion *DBSETCOL* innerhalb einer Dialogbox. "xl" und "yo" sind als Bytewerte anzugeben und bezeichnen die linke, obere Ecke des Rechtecks. "xb" und "yh" geben die Breite bzw. die Höhe des Rechtecks an.

"col" setzt sich aus der Zeichenfarbe (High-Nibble) und der Hintergrundfarbe (Low-Nibble) zusammen. Die Werte entsprechen den C64-Standardfarben.

Die Routine verändert wie die Routine *RecColorBox* den Akku, das x- und y-Register, sowie die Register *r5* bis *r8*.

4.32 i_UserColor (\$c0dc)

Die Routine *i_UserColor* entspricht der Routine *i_ColorBox* mit der Ausnahme, das der Farbwert nicht über die Inline-Daten übergeben wird, sondern direkt im Akku.

Diese Routine empfiehlt sich immer dann, wenn der Farbwert nicht fest vorgegeben ist, sondern aus einer Tabelle mit verschiedenen Farbwerten übernommen werden soll (z.B. aus der MP3-Farbtabelle).

```

lda    #$01          ; Schwarzer Text auf weißem Grund
jsr    i_UserColor
b xl    ; rel. x-Koordinate Links in Cards
b yo    ; rel. y-Koordinate Oben in Cards
b xb    ; Breite in Cards
b yh    ; Höhe in Cards
...
```

Verändert werden der Akku, das x- und y-Register, sowie die Register *r5* bis *r8*.

4.33 DirectColor (\$c0e2)

Diese Routine erstellt ebenfalls ein Rechteck mit frei wählbarer Farbe. *DirectColor* erwartet die y-Koordinate für den oberen Rand in *r2L*, die y-Koordinate für den unteren Rand in *r2H*, die linke x-Koordinate in *r3* und in *r4* die rechte x-Koordinate. Alle Werte werden in Pixeln angegeben. Die Belegung entspricht damit den Angaben der Routine *Rectangle*. Die Farbe wird im Akku übergeben. Beispiel:

```

:ClrColScrn    LoadB r2L,$00          ; Oberer Rand
                LoadB r2H,$c7          ; Unterer Rand
                LoadW r3,$0000          ; Linker Rand
                LoadW r4,$013f          ; Rechter Rand
                lda    #$02              ; Farbwert
                jsr    DirectColor
                ...
```

Der Vorteil von *DirectColor* gegenüber den Routinen *i_UserColor* oder *i_ColorBox* liegt in der Zusammenarbeit mit *Rectangle*. Hier noch ein Beispiel:

```

:drawColor     lda    #$03              ; Füllmuster setzen
                jsr    SetPattern

                jsr    i_Rectangle        ; Grafikrechteck zeichnen
                b      $08,$bf            ; Koordinaten werden nach
                w      $0008,$0137        ; r2L bis r4 kopiert

                lda    #$02              ; Farbrechteck zeichnen.
                jsr    DirectColor
                ...
```

Hier wird zuerst ein Rechteck mit dem Füllmuster \$03 gezeichnet. Die Koordinaten werden dabei von *i_Rectangle* nach *r2L* bis *r4* kopiert. *DirectColor* zeichnet dann mit diesen Koordinaten und dem Farbwert \$02 im Akku ein rotes Rechteck.

Die Werte in *r2L* bis *r4* enthalten die Koordinate des Rechtecks in Pixel, verändert werden Akku, x- und y-Register, sowie die Register *r5* bis *r8*.

4.34 GetBackScreen (\$c0e8)

Unter MP3 kann ein systemweit gültiges Hintergrundbild eingerichtet werden, welches von Anwendungen über diese Routine eingelesen werden kann. Ist kein Hintergrundbild eingerichtet, dann wird der Bildschirm.

GetBackScreen kopiert dazu die Hintergrundgrafik aus dem erweiterten RAM in den Vordergrundbildschirm des C64. Entscheidend hierfür ist das im Register *sysRAMFlg* das Bit 3 gesetzt ist. Ist Bit 3 nicht gesetzt, dann löscht diese Routine den Bildschirm mit einem vordefinierten Muster (enthalten im Register *BackScrPattern*).



Bild 2.2: Desktop-Oberfläche GeoDesk mit Hintergrundbild über *GetBackScreen*.

Die Routine verändert den Akku, x- und y-Register, sowie *r0* bis *r8* und *r11*.

4.35 ResetScreen (\$c0eb)

Diese Routine ist Teil der Routine *InitGEOS* und löscht den Bildschirm. Dabei werden außerdem die Standardfarben für Hintergrund und Rahmen gesetzt. Möchte man schnell den Grafikbildschirm löschen (Füllmuster \$02) und die Standardfarben von GEOS setzen (*C_GEOS_BACK*), dann kann diese Routine verwendet werden.

ResetScreen wird auch verwendet wenn kein Hintergrundbild aktiv ist.

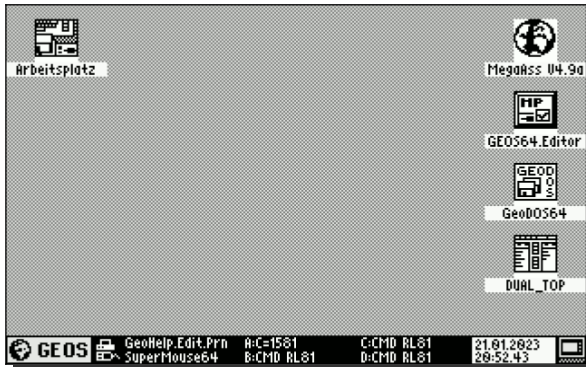


Bild 2.3: Desktop-Oberfläche GeoDesk ohne Hintergrundbild.

Die Routine verändert den Akku, x- und y-Register, sowie *r0* bis *r8* und *r11*.

Neue Diskettenroutinen (Ergänzung zu Teil B, Kapitel 12 ab Seite 282)

MP3 stellt auch eine Reihe an neuen Diskettenroutinen zur Verfügung. Dazu existiert ab \$9050 im Laufwerkstreiber eine erweiterte Sprungtabelle.

12.1 Die high-level-Diskettenroutinen

12.1.11 OpenRootDir (\$9050)

Die Routine *OpenRootDir* funktioniert nur in NativeMode-Laufwerkstreibern. Andere Treiber beenden diese Routine mit einem Fehler "ILLEGAL_DEVICE" (\$40).

OpenRootDir öffnet auf einem NativeMode-Laufwerk das Hauptverzeichnis und führt anschließend *OpenDisk* aus. Es werden keine besonderen Parameter benötigt. Nach dem Aufruf wird im x-Register die Fehlernummer übergeben (\$00=kein Fehler).

Verändert werden der Akku, x- und y-Register sowie die Adressen *r0* bis *r5*.

12.1.12 OpenSubDir (\$9053)

Öffnet auf NativeMode-Laufwerken ein Unterverzeichnis und führt anschließend *OpenDisk* aus. Dazu übergibt man in *r1L* den Track und in *r1H* der Sektor des ersten Verzeichnis-Blocks. Diese Angaben kann man direkt aus dem Verzeichnis-Eintrag entnehmen.



Hinweis:

Ab GEOS/MegaPatch V3.3r11 vom 14.3.2024 gibt es zusätzlich die Möglichkeit nur den Zeiger auf das aktuelle Verzeichnis zurückzusetzen. Dazu muss in *r1L* der Wert \$00 übergeben werden. Es wird danach kein *OpenDisk* ausgeführt!

Nach dem Aufruf wird im x-Register die Fehlernummer übergeben (\$00=kein Fehler). Bei einem anderen Fehler sollte *OpenRootDir* ausgeführt werden.

Verändert werden der Akku, x- und y-Register sowie die Adressen *r0* bis *r5*.

12.1.13 OpenPartition (\$9062)

Öffnet eine neue Partition auf dem aktuellen Laufwerk. Dazu übergibt man im Register *r3H* die gewünschte Partitions-Nr. Außerdem wird die Diskette mit *OpenDisk* geöffnet und die aktuelle BAM und der Diskettenname eingelesen.



Hinweis:

Ab GEOS/MegaPatch V3.3r11 vom 14.3.2024 öffnet *OpenPartition* auch das Hauptverzeichnis wenn ein ungültiges Unterverzeichnis aktiv ist. Es wird empfohlen zuvor *OpenRootDir* oder *OpenSubDir* mit *r1L*=\$00 aufzurufen.

Nach dem Aufruf wird im x-Register die Fehlernummer übergeben (\$00=kein Fehler).

Verändert werden der Akku, x- und y-Register sowie die Adressen *r0* bis *r5*.

12.1.14 GetPDirEntry (\$905c)

GetPDirEntry liest einen Eintrag aus dem Partitionsverzeichnis ein. Dazu übergibt man in *r3H* die Partitions-Nr. und in *r4* einen Zeiger auf einen 30 Byte großen Speicherbereich, der nach dem Aufruf die Partitionsdaten enthält.

Wichtig ist, dass diese Routine zu Beginn *ExitTurbo* und *InitForIO* aufruft und am Ende die Routine *DoneWithIO* aufgerufen wird. Ist der I/O-Bereich bereits aktiviert, dann muss die Routine *ReadPDirEntry* verwendet werden.

Nach dem Aufruf wird im x-Register die Fehlernummer übergeben (\$00=kein Fehler).

Innerhalb der Partitionsdaten findet man die Adresse des ersten Datenblock und die Partitionsgröße, diese werden in 512 Byte-Blocks ausgegeben, nicht wie sonst üblich als Blocks zu je 256 Byte! Damit lassen sich maximal 8Gb adressieren.

Die max. Partitionsgröße beträgt bei NativeMode etwas weniger als 16Mb (Spur 1-255 zu je 256 Blocks) bzw. 16Mb bei ForeignMode/DACC.

Verändert werden der Akku, x- und y-Register.

Hier eine Übersicht über die Angaben innerhalb der Partitionsdaten:

Adresse	Beschreibung
Byte 0	Partitionsformat: 0 - Nicht verwendet 1 - 1541 Emulation Mode 2 - 1571 Emulation Mode 3 - 1581 Emulation Mode 4 - Native Mode 5 - 1581 CP/M Emulation Mode 6 - Print Buffer 7 - ForeignMode / DACC 255 - Systempartition
Byte 1	\$00 (Reserviert)
Byte 2	Nr. der Partition, 1-254
Byte 3-18	Name der Partition, 16 Zeichen
Byte 19	Adresse erster 512-Byte-Datenblock (hb, Highbyte)
Byte 20	Adresse erster 512-Byte-Datenblock (mb, Middlebyte)
Byte 21	Adresse erster 512-Byte-Datenblock (lb, Lowbyte)
Byte 22-26	\$00 (Reserviert)
Byte 27	Partitionsgröße in 512-Byte-Blocks (hb, Highbyte)
Byte 28	Partitionsgröße in 512-Byte-Blocks (mb, Middlebyte)
Byte 29	Partitionsgröße in 512-Byte-Blocks (lb, Lowbyte)

Hinweis: Die Routine *GetPDirEntry* übergibt im Byte 0 der Partitionsdaten den Partitionstyp im GEOS-Format, nicht im CMD-Format!

Das CMD-DOS sendet im Anschluss an die 30 Byte Partitionsdaten noch ein CR(\$0d), das Byte wird aber von den Laufwerkstreibern überlesen. Damit kann als Speicherbereich für die Partitionsdaten auch *dirEntryBuf* (\$8400 bis \$841d, 30 Byte) verwendet werden.

12.1.15 GetPTypeData (\$9068)

Diese Routine erstellt eine Tabelle mit den Partitionstypen auf dem aktuellen Laufwerk. Wer z.B. wissen möchte welche Partitionen auf einem Laufwerk installiert sind, der kann diese Routine verwenden.

Zuvor muss man in r4 einen Zeiger auf einen 256-Byte großen Speicherbereich ablegen. Dort werden dann für die Partitionen 0-255 die Format-Byte abgelegt. Nach dem Aufruf wird im x-Register die Fehlernummer übergeben (\$00=kein Fehler).

12.2 Die mid-level-Diskettenroutinen

12.2.28 ReadPDirEntry (\$905f)

Wie *GetPDirEntry*, jedoch muss der Anwender zuvor die Routinen *ExitTurbo* und *InitForIO* manuell aufrufen. In *r3H* wird die Partitions-Nr. und in *r4* einen Zeiger auf einen 30 Byte großen Speicherbereich erwartet, der nach dem Aufruf die Daten der Partition enthält. Zum Schluss muss noch *DoneWithIO* aufgerufen werden.

Nach dem Aufruf von *ReadPDirEntry* enthält das x-Register die Fehlernummer (\$00=kein Fehler).

Verändert wird der Akku, das x- und y-Register.

12.2.29 SwapPartition (\$9065)

Wechselt die aktive Partition auf dem Laufwerk, zuvor müssen jedoch die Routinen *ExitTurbo* und *InitForIO* aufgerufen werden. Als Parameter übergibt man in *r3H* die Partitions-Nr. Zum Schluss muss noch *DoneWithIO* aufgerufen werden. Es wird anschließend kein *OpenDisk* ausgeführt. Auf NativeMode-Laufwerken muss zuvor die Routine *OpenRootDir* oder *OpenSubDir* mit *r1L*=\$00 ausgeführt werden, um den Zeiger auf das aktuelle Verzeichnis auf einen gültigen BAM-Block zurückzusetzen.

Diese Routine kann man in Zusammenhang mit *ReadPDirEntry* verwenden. Nach dem Aufruf wird im x-Register die Fehlernummer übergeben (\$00=kein Fehler).

Verändert werden der Akku, x- und y-Register sowie die Adressen *r0* bis *r5*.

12.3 Die low-level- und die very-low-level-Diskettenroutinen

12.3.12 GetBAMBlock (\$9056)

GetBAMBlock holt über *GetBlock* einen Block aus der aktuellen BAM nach *dir2Head*. Wurde die BAM im aktuellen Block verändert, dann wird der aktuelle Block über *PutBAMBlock* gespeichert bevor der neue Block eingelesen wird.

Die Routine hat für Programmierer keine Bedeutung, da diese nur von den internen BAM-Routinen verwendet wird.

Als einziger Parameter übergibt man im Akku die Block-Adresse des BAM-Blocks (Werte von 2 bis 33). Bei einem Fehler wird im x-Register die Fehlernummer übergeben (\$00=kein Fehler).

Verändert wird der Akku, das x- und y-Register.

12.3.13 PutBAMBlock (\$9059)

Ähnlich *GetBAMBlock*, aber der angegebene BAM-Block wird auf Diskette gespeichert. Als Parameter wird im Akku die Block-Adresse des BAM-Blocks (Werte von 2 bis 33) erwartet. Nach dem Aufruf enthält das x-Register die Fehlernummer (\$00=kein Fehler).

Da außer der CMD-HD kein anderes Laufwerk bis zu 255 Partitionen unterstützt, werden die restlichen Bytes mit *NULL*-Byte aufgefüllt. Zu beachten ist, das Format-Byte im GEOS-Format und nicht im CMD-Format abgelegt werden.

Verändert werden der Akku, x- und y-Register.

12.3.14 SendFloppyCom (\$906b)

Diese Routine sendet einen Floppy-Befehl an das entsprechende Laufwerk. Auf RAM-Laufwerken erhält man hier einen Fehler "ILLEGAL_DEVICE" (\$40).

Vor dem Aufruf muss in *r0* ein Zeiger auf den Laufwerksbefehl abgelegt werden. Die Länge des Befehls muss im Register *r2L* abgelegt werden. Hier ist zu beachten, dass die Länge des Floppy-Befehls nicht beliebig groß sein darf, ca. 40 Bytes sollten aber, je nach Laufwerk, möglich sein.

Vorher müssen jedoch noch die Routinen *ExitTurbo* und *InitForIO* und am Ende dann wieder *DoneWithIO* aufgerufen werden. Nach dem Aufruf von *SendFloppyCom* enthält das x-Register die Fehlernummer (\$00=kein Fehler).

Verändert wird der Akku, das x- und y-Register.



ACHTUNG! Die folgenden Routinen gelten nur für GEOS/MegaPatch-Laufwerkstreiber, die über die Formatkennung "DDX" (*DiskDrvExtType*, \$907a) verfügen, siehe **Teil D, Kapitel 2.2 ab Seite 466**.

12.3.15 InitForDDrvOp (\$907c)

Diese Routine setzt die Speicheradressen für den aktuellen Laufwerkstreiber im C64- und im erweiterten GEOS-Speicher. Danach kann entweder die Routine *FetchRAM* oder *StashRAM* verwendet werden, um den Treiber auszulesen oder die Werte für *DDRV_EXT_DATA1* (\$907a) bzw. *DDRV_EXT_DATA2* (\$907b) im erweiterten GEOS-Speicher zu sichern.

Nach dem Aufruf findet man in *r0* die Adresse des Laufwerkstreiber im RAM, in *r1* die Adresse in der Speichererweiterung. Die Größe in *r2* beträgt immer \$0d80 Byte und die Speicherbank in *r3L* ist immer Bank#0.

Verändert wird der Akku, das x- und y-Register sowie die Register *r0* bis *r3L*.

12.3.16 DoneWithDDrvOp (\$907f)

Nach *InitForDDrvOp* und *StashRAM/FetchRAM* muss auch immer *DoneWithDDrvOp* aufgerufen werden, da die Register *r0* bis *r3L* im GEOS-Kernal zwischengespeichert und mit den Adressen des Laufwerkstreibers getauscht werden. Diese Routine setzt daher am Ende die Werte in *r0* bis *r3L* wieder auf die Anfangswerte zurück.

Verändert wird der Akku, das x- und y-Register sowie die Register *r0* bis *r3L*.

Eine Routine zum permanenten speichern der Werte in *DDRV_EXT_DATA1* (\$907a) bzw. *DDRV_EXT_DATA2* (\$907b) im GEOS-DACC könnte wie folgt aussehen:

```
:UpdateGEOS    jsr    InitForDDrvOp    ; Zeiger setzen
                jsr    StashRAM         ; DDX-Register speichern
                jsr    DoneWithDDrvOp   ; Zeiger zurücksetzen.
```

Hinweis: Einige der neuen Register werden nur zum Teil automatisch im Treiber gespeichert, z.B. *Flag_SD2IEC* oder *GeoRAMBSize*.

Die reservierten Adressen *DDRV_EXT_DATA1* und *DDRV_EXT_DATA2* können zwar von Anwendungsprogrammen genutzt werden, der GEOS-Kernal aktualisiert die Werte aber nicht automatisch im erweiterten GEOS-Speicher.

Neue Systemroutinen (Ergänzung zu Teil B, Kapitel 14 ab Seite 310)

14.12 GEOS_InitSystem (\$c0ee)

Setzt alle GEOS-Register und Systemvektoren auf Standardwerte zurück. Wird z.B. von GEOS beim starten einer Application über *GetFile* verwendet.



ACHTUNG!

Diese Routine ist Teil der GEOS-Initialisierung und verändert auch Werte im I/O-Bereich ab \$d000, daher muss vor dem Aufruf der Routine der Interrupt gesperrt werden. Dazu kann folgendes Beispiel verwendet werden:

```
:doInitGEOS    php           ; Interrupt-Status speichern
                sei           ; Interrupt sperren
                jsr    GEOS_InitSystem ; GEOS-Werte zurücksetzen
                plp           ; Interrupt-Status zurücksetzen
```

Die Routine verändert Akku, x- und y-Register, sowie die Register *r0* bis *r2L*.

14.13 PutKeyInBuffer (\$c0f1)

Der C64-BASIC-Editor besitzt die Möglichkeit des "programmierten Direktmodus". Hierbei werden in den Tastaturpuffer von GEOS Tastencodes eingetragen, welche dann erst später ausgeführt werden.

Die Routine *PutKeyInBuffer* kopiert einen Tastencode in den Tastaturpuffer, der dann vom Kernl nach *keyData* kopiert wird. Ein Anwendungsbeispiel wäre z.B. bei einer Tastatureingabe über *GetString* den Mausabfrage-Vektor *otherPressVec* auf folgende Routine zu setzen:

```
:ExGetStrg     lda    #KEY_CR
                jmp    PutKeyInBuffer
```

Wenn der Anwender während der Texteingabe jetzt einen Maustaste oder der Feuerknopf drückt, dann wird diese Routine ausgeführt und der Tastencode *CR* (\$0d; **[RETURN]**-Taste) in den Tastaturpuffer geschrieben. Beim nächsten Durchlauf der Mainloop wird dann die Texteingabe beendet.

Die Routine wird z.B. vom RegisterMenü verwendet, wenn der Anwender bei einer aktiven Tastatureingabe mit der Maustaste / dem Joystick-Button klickt: In dem Fall wird die Eingabe automatisch beendet.

Verändert wird der Akku und das x-Register. Verändert wird außerdem *pressFlag*.



ACHTUNG! Die Routine funktioniert nur mit dem Original GEOS/MegaPatch-Kernal. Wird GEOS/MegaPatch für die PC-Tastatur mit "geoKeys" angepasst, dann erzeugt die Routine keine Reaktion, da Tastatureingaben vom PC-Tastatortreiber auf andere Art und Weise verwaltet werden!

14.14 SCPU_Pause (\$c0f4)

Häufig ist es notwendig den C64 eine genau definierte Zeit "warten" zu lassen. Die sonst üblichen Schleifen können hier nicht immer verwendet werden.

```

::loop      ldx    #$80          ; Max. 128 Durchläufe
            dex     ; Schleife beendet ?
            bne     :loop       ; Nein, weiter...
```

Diese Schleifen haben den Nachteil, das sie auf unterschiedlichen Systemen unterschiedlich schnell abgearbeitet werden. Auf einem C128 (im 80-Zeichen-Modus) ist eine solche Schleife doppelt so schnell wie auf einem C64, auf einem System mit einer CMD-SuperCPU sogar bis zu 20x schneller.

Die Routine *SCPU_Pause* wartet hier immer circa eine 1/10-Sekunde. Erreicht wird das durch den Vergleich mit der Systemzeit des Computers (1/10-Sekundenregister im CIA#1;\$dc08). Im VICE-Emulator funktioniert die Routine im WARP-Modus nicht.

Verändert wird nur der Inhalt des Akku.

14.15 SCPU_OptOn (\$c0f7)

Aktiviert die Optimierung für GEOS auf SuperCPU-Systemen und ist standardmäßig eingeschaltet. Dabei wird das SuperCPU-Hardware-Register \$d074 beschrieben, was die Optimierung des GEOS-Speichers von \$8000-\$bfff (VIC-Speicherbank 2) aktiviert. Der Speicherbereich wird dann im SuperCPU-RAM gespiegelt, da Speicherzugriffe hier schneller ausgeführt werden können.

Ist keine CMD-SuperCPU vorhanden, dann bleibt dieser Aufruf ohne Funktion.

Verändert wird der Inhalt von Akku, x- und y-Register. Außerdem wird der Wert in *Flag_Optimize* auf \$00 gesetzt.

14.16 SCPU_OptOff (\$c0fa)

Deaktiviert die Optimierung für GEOS auf CMD-SuperCPU-Systemen.

Das kann notwendig werden, wenn z.B. in den Textmodus geschaltet werden soll (wie etwa beim Programm *geoDebugger* oder *geoBasic*). Dazu wird das SuperCPU-Hardware-Register \$d077 beschrieben und die VIC-Speicherbank 2 von \$8000-\$bfff wird wieder im Computer-RAM aktiviert.

Ist keine CMD-SuperCPU vorhanden bleibt dieser Aufruf ohne Funktion.

Verändert wird der Inhalt von Akku, x- und y-Register. Außerdem wird der Wert in *Flag_Optimize* auf \$03 gesetzt.

14.17 SCPU_SetOpt (\$c0fd)

Setzt oder löscht die GEOS-Optimierung für die CMD-SuperCPU. Wenn die Optimierung eingeschaltet werden soll, dann muss im Register *Flag_Optimize* der Wert \$00 eingetragen werden. Soll die Optimierung abgeschaltet werden, dann muss *Flag_Optimize* auf den Wert \$03 gesetzt werden.

Ist keine CMD-SuperCPU vorhanden bleibt dieser Aufruf ohne Funktion.

Verändert wird der Inhalt von Akku, x- und y-Register.

2.7 Das Registermenü

Das Registermenü stellt eine neue Form von Menüs unter GEOS dar.

Im Gegensatz zu *DoMenu* und *Dolcons* können hier auch Zahlen und Texte eingegeben werden, Listen verwaltet oder Optionen aktiviert/deaktiviert werden. Das Registermenü stellt dazu Kartenreiter zur Verfügung, über die man zwischen verschiedenen Untermenüs wechseln kann. Damit lassen sich auch komplexere Menüs übersichtlich gestalten.

Für ein Registermenü übergibt man in *r0* einen Zeiger auf eine Registertabelle, ähnlich zur Routine *DoMenu*. Vorher muss jedoch die Registermenü-Routine in den Speicher geholt werden. Dazu kann man folgende Routine verwenden:

```
:LdRegMenu    jsr    SetADDR_Register ; Zeiger für FetchRAM setzen
              jsr    FetchRAM         ; Register-Menü einlesen
              ...
```

Das Registermenü liegt im *APP_RAM* von \$6d00 bis \$78ff im Speicher, daher dürfen Anwendungen jetzt nur noch den Bereich von \$0400 bis \$6cff belegen. Der Speicher im Bereich des Registermenüs wird nicht zwischengespeichert. Sofern hier Programmdaten liegen muss die Anwendung den Bereich eigenständig sichern.

Hier nun der grundlegende Aufbau eines Registermenüs:

```
:RMenuTab      b yoben                ; y-Koordinate oben
               b yunten               ; y-Koordinate unten
               w xlinks               ; x-Koordinate links
               w xrechts              ; x-Koordinate rechts

               b Anzahl Register      ; Anzahl der Registerkarten

               w RNmIcon01            ; Zeiger auf Registericon
               w RTabDat01            ; Zeiger auf Registerdaten

               w RNmIcon02            ; Zeiger auf Registericon
               w RTabDat02            ; Zeiger auf Registerdaten

               ...                    ; Ggf. weitere Registerkarten

:RNmIcon01     w RRIcon01             ; Zeiger auf Icon für Registerkarte
               b xpos,ypos,breite,hoehe ; Iconposition und -größe

:RNmIcon02     w RRIcon02             ; Zeiger auf Icon für Registerkarte
               b xpos,ypos,breite,hoehe ; Iconposition und -größe
               ...

:RTabDat01     b Anzahl Einträge       ; Anzahl Einträge auf Registerkarte

               b Typ Register-Eintrag ; Registertyp
               (10 Byte Systemdaten)  ; Daten für Registertyp

               b Typ Register-Eintrag ; Registertyp
               (10 Byte Systemdaten)  ; Daten für Registertyp

               ...                    ; Ggf. weitere Register-Einträge
```

Die ersten sechs Byte bestimmen die Größe der Registerkarten, die Größenangabe entspricht dem Format von *i_Rectangle*. Es ist dabei zu beachten, das die Kartenreiter in der Größenangabe nicht mit enthalten sind. Die größte Ausdehnung für ein Registermenü wäre demnach:

```
:RMenTab      b $08                ; y-Koordinate oben
               b $c7                ; y-Koordinate unten
               w $0000              ; x-Koordinate links
               w $013f              ; x-Koordinate rechts
```

Werden mehrere Registerkarten verwendet, dann können die Kartenreiter auch zweizeilig angeordnet werden. Jede weitere Zeile mit Kartenreitern verringert die vertikale Größe des Registermenüs um 8 Pixel.

Die Größe sollte außerdem auch auf den Anfang (links, oben) oder das Ende (rechts, unten) ausgerichtet werden, da die Registerkarten in Farbe dargestellt werden.

Da Farbe nur innerhalb von ganzen Cards gesetzt werden kann sollte das bei der Größe eines Registermenüs beachtet werden. Es kann daher Hilfreich sein, wenn man sich zu Beginn des Projektes eine Skizze des Registermenüs auf Karopapier erstellt.

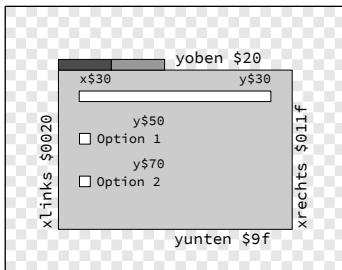


Bild 2.2:
Skizze für ein Registermenü

Danach folgt die Anzahl der Registerkarten. Zu der Anzahl der Registerkarten muss nun die passende Anzahl an Registerverweisen folgen. Ein Registerverweis besteht aus einem Word für einen Zeiger auf das Kartenreiter-Icon und einem Word für die Definitionstabelle mit den Daten für diese Registerkarte.

Der Eintrag für das Kartenreiter-Icon ähnelt dem Eintrag in der Tabelle von *Dolcons*. Als erstes wird ein Zeiger (Word) auf die Icon-Grafik erwartet. Danach folgt die horizontale Position (xpos) in Cards, gefolgt von der vertikalen Position (ypos) in Pixel, die Breite wird in Cards angegeben, die Höhe wiederum in Pixel.

Breite und Höhe des Icon können über die beiden Pseudo-Labels *.x* und *.y* definiert werden. Eine Eintrag für einen Kartenreiter könnte dann so aussehen:

```
:RNMIcon01    w RRIcon01            ; Zeiger auf Icondaten
               b $10,$20            ; x-/y-Koordinate
               b RRIcon01_x, RRIcon01_y; Breite und Höhe des Icon

:RRIcon01      [DESKTOP]

:RRIcon01_x    = .x
:RRIcon01_y    = .y
```

Für ein Registermenü stehen verschiedene Datentypen zur Verfügung um die Inhalte einer Registerkarte zu gestalten.

Funktionsblock

Typ	Bezeichnung	Bedeutung	Parameter
1	BOX_USER	Menübereich	Word, Zeiger auf Titel Word, Zeiger auf Anwenderroutine Byte, y-Koord. in Pixel, obere Grenze Byte, y-Koord. in Pixel, untere Grenze Word, x-Koord. in Pixel, linke Grenze Word, x-Koord. in Pixel, rechte Grenze
2	BOX_USER_VIEW	Menübereich (Nur anzeigen)	Word, Zeiger auf Titel Word, Zeiger auf Anwenderroutine Byte, y-Koord. in Pixel, obere Grenze Byte, y-Koord. in Pixel, untere Grenze Word, x-Koord. in Pixel, linke Grenze Word, x-Koord. in Pixel, rechte Grenze
3	BOX_USEROPT	Menübereich mit Optionsfeld	Word, Zeiger auf Titel Word, Zeiger auf Anwenderroutine Byte, y-Koord. in Pixel, obere Grenze Byte, y-Koord. in Pixel, untere Grenze Word, x-Koord. in Pixel, linke Grenze Word, x-Koord. in Pixel, rechte Grenze
4	BOX_USEROPT_VIEW	Menübereich mit Optionsfeld (Nur anzeigen)	Word, Zeiger auf Titel Word, Zeiger auf Anwenderroutine Byte, y-Koord. in Pixel, obere Grenze Byte, y-Koord. in Pixel, untere Grenze Word, x-Koord. in Pixel, linke Grenze Word, x-Koord. in Pixel, rechte Grenze
5	BOX_FRAME	Gruppenrahmen	Word, Zeiger auf Titel Word, Zeiger auf Anwenderroutine Byte, y-Koord. in Pixel, obere Grenze Byte, y-Koord. in Pixel, untere Grenze Word, x-Koord. in Pixel, linke Grenze Word, x-Koord. in Pixel, rechte Grenze
6	BOX_ICON	Icon-Option	Word, Zeiger auf Titel Word, Zeiger auf Anwenderroutine Byte, y-Koord. in Pixel Word, x-Koord. in Pixel Word, Zeiger auf Icon-Tabelle Byte, Registereintrag oder \$00
7	BOX_ICON_VIEW	Icon-Option (Nur anzeigen)	Word, Zeiger auf Titel Word, Zeiger auf Anwenderroutine Byte, y-Koord. in Pixel Word, x-Koord. in Pixel Word, Zeiger auf Icon-Tabelle Byte, \$00

Typ	Bezeichnung	Bedeutung	Parameter
8	BOX_OPTION	Optionsfeld	Word, Zeiger auf Titel Word, Zeiger auf Anwenderroutine Byte, y-Koord. in Pixel Word, x-Koord. in Pixel Word, Zeiger auf Datenbyte Byte, Bit-Maske
9	BOX_OPTION_VIEW	Optionsfeld (Nur anzeigen)	Word, Zeiger auf Titel Word, Zeiger auf Anwenderroutine Byte, y-Koord. in Pixel Word, x-Koord. in Pixel Word, Zeiger auf Datenbyte Byte, Bit-Maske
10	BOX_STRING	Textfeld	Word, Zeiger auf Titel Word, Zeiger auf Anwenderroutine Byte, y-Koord. in Pixel Word, x-Koord. in Pixel Word, Zeiger auf Textstring Byte, Anzahl Zeichen
11	BOX_STRING_VIEW	Textfeld (Nur anzeigen)	Word, Zeiger auf Titel Word, Zeiger auf Anwenderroutine Byte, y-Koord. in Pixel Word, x-Koord. in Pixel Word, Zeiger auf Textstring Byte, Anzahl Zeichen
12	BOX_NUMERIC	Zahlenfeld	Word, Zeiger auf Titel Word, Zeiger auf Anwenderroutine Byte, y-Koord. in Pixel Word, x-Koord. in Pixel Word, Zeiger auf Zahlenwert Byte, Anzahl Ziffern
13	BOX_NUMERIC_VIEW	Zahlenfeld (Nur anzeigen)	Word, Zeiger auf Titel Word, Zeiger auf Anwenderroutine Byte, y-Koord. in Pixel Word, x-Koord. in Pixel Word, Zeiger auf Zahlenwert Byte, Anzahl Ziffern

Alle Datentypen haben gemeinsam, das im ersten Word ein Zeiger auf einen Titel für das Optionsfeld erwartet wird. Der Text muss mit einer Koordinatenangabe beginnen (mit Ausnahme von *BOX_FRAME*), analog zu den Inline-Daten von *i_PutString*:

```
:TITEL1      w xpos           ; x-Koordinate
              b ypos           ; y-Koordinate, Grundlinie!
              b "Titeltext",NULL ; Titeltext, NULL als Ende-Kennung
```

Wenn kein Titel angezeigt werden soll, dann kann hier \$0000 angegeben werden.

Nach dem Zeiger auf den Titel folgt immer ein Zeiger auf eine Anwenderroutine, die entweder Daten ausgibt oder eine Aktion ausführt. Auch hier kann der Wert \$0000 angegeben werden, wenn keine Routine ausgeführt werden soll.

BOX_USER (\$01)

BOX_USER definiert einen Bereich auf dem Bildschirm, der mit dem Mauszeiger angeklickt werden kann. Nach einem Mausklick wird eine Anwenderoutine aufgerufen, die den Mausklick auswerten kann. Beispiel:

```

b BOX_USER          ; Datentyp
w TITEL             ; Zeiger auf optionalen Titeltext
w ROUTINE           ; Anwenderoutine
b yoben             ; y-Koordinate oben
b yunten            ; y-Koordinate unten
w xlinks            ; x-Koordinate links
w xrechts           ; x-Koordinate rechts

```

»TITEL« ist ein Zeiger auf einen Text, welcher den Bereich auf dem Bildschirm beschreibt. Da der Text nicht an einer bestimmten Stelle angezeigt wird, muss der Text mit einem Word für die x-Koordinate und einem Byte für die y-Koordinate beginnen. Soll kein Text ausgegeben werden, dann ist »TITEL« gleich \$0000.

»ROUTINE« zeigt auf eine Anwenderoutine, die zwei Aufgaben erfüllt: Während das Registermenü aufgebaut wird, muss diese Routine den Inhalt des Bereichs zeichnen. Wenn das Menü aktiv ist und der Anwender den Bereich mit dem Mauszeiger angeklickt hat, dann muss diese Routine auch den Mausklick auswerten.

Um nun unterscheiden zu können wann die Routine aufgerufen wurde, kann man das Register *r1L* abfragen: Findet man in *r1L* den Wert \$00, dann wird das Register aufgebaut. Der Wert \$ff in *r1L* dagegen bedeutet, das der Bereich mit der Maus angeklickt und das Programm soll den Mausklick auswerten.

Die Koordinaten yoben, yunten, xlinks und xrechts bestimmen die Größe und beziehen sich auf den gesamten Grafikbildschirm!

Man verwendet hier also keine relativen Koordinaten. Um sich die Arbeit mit der Positionierung der Einträge zu erleichtern kann man sich mit einem Trick behelfen: Man definiert Labels für die linke und obere Grenze des Registermenüs und bezieht dann die Koordinaten relativ zu diesen Labels.

```

:RMOben      = $20          ; Oberer Rand
:RMUnten     = $9f          ; Unterer Rand
:RMLinks     = $0020        ; Linker Rand
:RMRechts    = $011f        ; Rechter Rand

:RMenuTab
b RMOben      ; y-Koordinate oben
b RMUnten     ; y-Koordinate unten
w RMLinks     ; x-Koordinate links
w RMRechts    ; x-Koordinate rechts

b BOX_USER    ; Datentyp
w TITEL       ; Zeiger auf optionalen Titeltext
w ROUTINE     ; Anwenderoutine
b RMOben +$10 ; y-Koordinate oben
b RMOben +$2f ; y-Koordinate unten
w RMLinks +$0010 ; x-Koordinate links
w RMLinks +$007f ; x-Koordinate rechts

```

Damit sind die Koordinatenangaben "relativ" zum Rand der Registerkarte. Sollen mehrere Einträge im Menü unter- oder nebeneinander angeordnet werden, dann sind Änderungen relativ aufwändig, da man alle Zahlen von Hand anpassen muss. Hier kann man zu einem weiteren Trick greifen: Horizontale und vertikale Tabulatoren:

```
:RMOben      = $20           ; Oberer Rand
:RMUnten     = $9f           ; Unterer Rand
:RMLinks     = $0020         ; Linker Rand
:RMRechts    = $011f         ; Rechter Rand

:RTY1        = $10           ; Vertikaler Tabulator
:RTX1        = $0010         ; Horizontaler Tabulator
:RTX2        = $00a0         ; Zweiter horizontaler Tabulator

:RMenuTab    b RMOben        ; y-Koordinate oben
               b RMUnten      ; y-Koordinate unten
               w RMLinks      ; x-Koordinate links
               w RMRechts     ; x-Koordinate rechts
               ...

               b BOX_USER     ; Datentyp
               w TITEL        ; Zeiger auf optionalen Titeltext
               w ROUTINE      ; Anwenderroutine
               b RMOben +RTY1 ; y-Koordinate oben
               b RMOben +RTY1 +$1f ; y-Koordinate unten
               w RMLinks +RTX1 ; x-Koordinate links
               w RMLinks +RTX1 +$006f ; x-Koordinate rechts
               ...
```

Damit wird bei der Größe des Bereichs nur noch die Höhe und die Breite absolut angegeben. Wenn man Einträge innerhalb der Registerkarte verschieben muss, dann reicht es aus die Tabulatoren anzupassen. Alle Einträge, die den gleichen Tabulator verwenden, werden dann automatisch verschoben.

Zur Veranschaulichung hier noch eine Beispielgrafik:

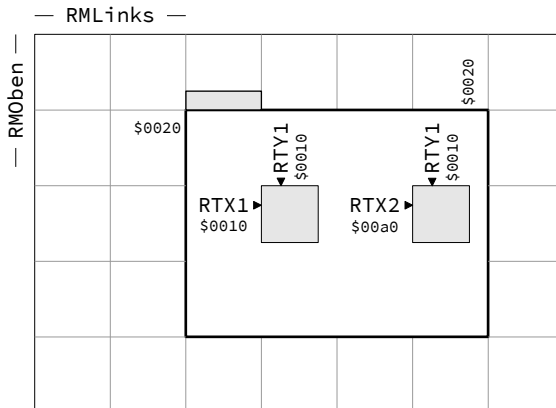


Bild 2.3: Horizontale und vertikale Tabulatoren

BOX_USER_VIEW (\$02)

Dieser Funktionscode entspricht mit einer Ausnahme dem Typ *BOX_USER*: Die Anwenderroutine kann nur mit dem Wert \$00 in *r1L* aufgerufen werden. Ein anklicken mit der Maus ist nicht möglich. Man sollte daher den Wert in *r1L* nicht auswerten.

Ein Eintrag vom Typ *BOX_USER_VIEW* kann z.B. dazu verwendet werden, einen Eintrag kurzfristig zu deaktivieren. Dazu ersetzt man in der Registertabelle den Datentyp *BOX_USER* durch *BOX_USER_VIEW*. Das kann auch geschehen wenn das Registermenü bereits aktiv ist, da bei jedem Mausklick die Registertabelle vollständig abgearbeitet wird.

Es ist allerdings möglich, den Eintrag über den Datentyp *BOX_ICON* auf dem Bildschirm zu aktualisieren. Wie das funktioniert wird später noch im Detail erklärt.

BOX_USEROPT (\$03)

In der Funktion identisch mit *BOX_USER*. Hier wird jedoch zusätzlich ein Optionsfeld in der Größe der angegebenen Koordinaten auf der Registerkarte gezeichnet. Es wird dabei ein Rahmen gezeichnet, der mit einer Farbfläche ausgefüllt ist. Die Farbe für das Optionsfeld ist in MP3 über *C_InputField* definiert.

BOX_USEROPT_VIEW (\$04)

Wie *BOX_USEROPT* zeichnet auch dieser Datentyp ein Optionsfeld auf der Registerkarte und erlaubt nur die Darstellung von Informationen beim Aufbau des Registermenüs. Eine anklicken mit der Maus führt zu keiner Reaktion.

BOX_FRAME (\$05)

Der Befehl *BOX_FRAME* dient zum Einteilen der Registerkarte in verschiedene Bereiche und zeichnet einen Frame (Rahmen) mit einem Titeltex.

Nach dem Funktionscode folgt ein Word als Zeiger auf den Titeltex. Dieser wird nicht mit einer Koordinatenangabe begonnen, da der Titel immer am linken/oberen Rand des Rahmens dargestellt wird.

Soll kein Titeltex angezeigt werden, dann setzt man den Zeiger auf den Wert \$0000.

Der Rahmen führt keine Benutzeraktionen aus, die Anwenderroutine kann daher nur dazu verwendet werden, während dem Aufbau des Registermenüs verschiedene Inhalte auf dem Bildschirm auszugeben.

BOX_ICON (\$06)

Der Datentyp *BOX_ICON* zeichnet ein Icon auf der Registerkarte, welches beim anklicken invertiert wird oder einmalig aufblinkt.

Nach den beiden Word für den Titel und die Anwenderroutine folgt die y- und x-Koordinate des Icon auf dem Bildschirm. Beide Werte werden in Pixel angegeben. Anschließend folgt ein Zeiger (Word) auf eine Icon-Tabelle und ein Optionsbyte. Hier ein Beispiel:

```
b BOX_Icon          ; Datentyp
w TITEL             ; Zeiger auf optionalen Titeltex
w ROUTINE           ; Anwenderroutine
b RMOben +$10       ; y-Koordinate
w RMLinks +$0010    ; x-Koordinate
w IconEntry         ; Zeiger auf Icon-Tabelle
b NO_OPT_UPDATE     ; Optionsbyte
```


Die Koordinate sollte wegen der Farbe auf ganze Cards ausgerichtet sein. »:IconEntry« verweist auf eine Icon-Tabelle, in der Angaben zur Icon-Grafik, Breite und Höhe des Icon, und zur Farbe des Icon enthalten sind.

Das letzte Byte im Definitionsblock zeigt auf einen Eintrag in der Registertabelle, der aktualisiert werden muss, wenn die Anwenderroutine ausgeführt wurde. Wenn kein anderer Eintrag aktualisiert werden muss, dann steht hier *NO_OPT_UPDATE* (\$00).

Zu beachten ist dabei, dass der erste Eintrag in Registertabelle die Nr.1 hat! Damit beim Einfügen von Einträgen der Wert nicht manuell angepasst werden muss, hilft hier wieder ein kleiner Trick:

```
:RTabDat01      b Anzahl Einträge          ; Anzahl Einträge auf Registerkarte
:u01            b BOX_NUMERIC              ; Erste Option: Zahleneingabe
                w $0000
                w $0000
                b RMOben +RTY1
                w RMLinks +RTX1
                w DATAWORD
                b 4 ! NUMERIC_LEFT ! NUMERIC_BYTE

                b BOX_ICON                  ; Zweite Option: Icon-Option
                w $0000
                w $0000
                b RMOben +RTY2
                w RMLinks +RTX2
                w IconEntry
                b (:u01 -RDataTab01 -1)/11 +1
```

Der Eintrag, der aktualisiert werden soll, erhält ein lokales Label, hier »:u01«. Beim Eintrag für *BOX_ICON* findet dann beim Optionsbyte eine Berechnung statt.

Da alle Einträge genau 11 Byte umfassen, muss nur der Abstand zwischen dem *BOX_ICON* und dem Tabellenanfang ermittelt und durch 11 dividiert werden. Es muss lediglich das Kopfbyte mit den Anzahl der Einträge berücksichtigt werden (-1) und anschließend der Wert auf den eigentlichen Einträge korrigiert werden (+1).

Wie sieht nun eine solche Icon-Tabelle aus? Hier ein Beispiel:

```
:IconEntry      w IconGrafik                ; Zeiger auf Bitmap für das Icon
::opt           b %11000000                ; Bit 7=1: ":iconSelFlag" beachten
                                           ; Bit 6=1: Icon nicht invertieren
                b $00                      ; Dummy-Byte
                b IconGrafik_x              ; Breite des Icon
                b IconGrafik_y              ; Höhe des Icon
                b Farbwert                  ; Farbe für das Icon
```

Die beiden Byte ab dem Label »::opt« waren ursprünglich für die Koordinaten des Icon reserviert, analog zu *DBUSRICON* von *DoDigBox*. Aktuell wird nur Bit 6+7 zu besserer Sichtbarkeit von angeklickten Icons verwendet.

Die Icon-Größe kann über *.x* (Breite / Cards) und *.y* (Höhe / Pixel) definiert werden.

Beim Farbwert gilt das bereits bekannte: Der Wert setzt sich aus der Zeichenfarbe (High-Nibble) und der Hintergrundfarbe (Low-Nibble) zusammen. Die einzelnen Werte entsprechen den C64-Standardfarben.

Beim Farbwert gibt es allerdings zwei Werte mit besonderen Eigenschaften *USE_COLOR_INPUT* (\$ff) und *USE_COLOR_REG* (\$ee):

Der Farbwert *USE_COLOR_INPUT* deutet an, das für das Icon der Farbwert für ein Eingabefeld verwendet werden soll. Der Wert ist also ein Platzhalter, der vom Registermenü automatisch mit der Farbe aus *C_InputField* ersetzt wird.

Die zweite Ausnahme ist *USE_COLOR_REG*, bei der als Farbwert die Farbe der Registerkarten verwendet wird. Dieser Wert ist auch ein Platzhalter und wird vom Registermenü durch die Farbe für die Registerkarten aus *C_RegisterBack* ersetzt.

USE_COLOR_INPUT verwendet man, wenn ein Icon vom Anwender angeklickt werden kann, *USE_COLOR_REG* wird sinnvollerweise für ein Icon verwendet, das nur zur Information oder zur Dekoration dient.

Die Werte \$ee und \$ff wurden gewählt, weil diese Farben nur selten Verwendung finden dürften: Vorder- und Hintergrund haben hier denselben Farbwert.

BOX_ICON_VIEW (\$07)

Ähnlich wie *BOX_ICON* zeichnet diese Routine ein Icon auf den Bildschirm. Das Icon kann aber nicht mit der Maus angeklickt werden.

Eine Verwendungsmöglichkeit wäre ein *BOX_ICON* temporär abzuschalten, indem man den Datentyp auf *BOX_ICON_VIEW* ändert. Das Icon ist dann zwar noch sichtbar, ein Mausklick auf das Icon führt aber keine Aktion mehr aus.

BOX_OPTION (\$08)

Mit *BOX_OPTION* ist es möglich Optionen ein- oder auszuschalten. Dabei werden in einem definierten Datenbyte mit Hilfe einer Bit-Maske einzelne Bits invertiert.

Dazu wird bei jedem *BOX_OPTION* ein kleines Optionsfeld auf dem Bildschirm angezeigt, welches der Anwender mit der Maus anklicken kann. Eine aktivierte Option wird dann mit einem Haken markiert. Hier ein Beispiel:

```

b BOX_OPTION           ; Optionsfeld
w TITEL                ; Zeiger auf Titel für Option
w ROUTINE              ; Anwenderroutine
b RMOben +RTY1         ; y-Koordinate
w RMLinks +RTX1        ; x-Koordinate
w DATABYTE             ; Zeiger auf Datenbyte
b %1000 0000          ; Bit-Maske

```

»TITEL« und »ROUTINE« dürften bereits bekannt sein. Danach folgt die Koordinate für die linke, obere Ecke des Optionsfeldes. Die Koordinate sollte wegen der Farbe auf ganze Cards ausgerichtet sein.

»DATABYTE« ist ein Zeiger auf eine Adresse, in der sich der Optionswert befindet. Der Optionswert ist aber keine Zahl, sondern eher eine Liste von Bit, die ein- oder ausgeschaltet werden können.

Das letzte Byte definiert dann die Bit-Maske, die über den Befehl **eor** den Optionswert verändert. Wenn beim anklicken der Option nur Bit 7 invertiert werden soll, dann ist hier der Wert %1000 0000 anzugeben.

Sollen alle Bit invertiert werden, dann steht hier der Wert %1111 1111 oder \$ff. In dem Fall wird der Optionswert dann nur zwischen den beiden Zuständen \$00 und \$ff umgeschaltet.

Was passiert nun wenn der Anwender das Optionsfeld mit der Maus anklickt?

Zuerst ändert die Registerroutine die entsprechenden Bit im Optionswert, welche mit der Bit-Maske definiert wurden. Anschließend wird die Anwenderoutine aufgerufen.

Hier kann das Programm jetzt überprüfen, ob das ändern der Option überhaupt zulässig war. Falls der Wert nicht gültig ist, dann können die einzelnen Bits wieder zurückgesetzt oder auf einen gültigen Wert angepasst werden.

Im Anschluss prüft dann die Registerroutine, ob die entsprechenden Bit gemäß der Bit-Maske im Optionswert gesetzt sind. Wenn die Bit gesetzt sind, dann wird die Option als "Aktiv", also mit einem Haken, markiert.

BOX_OPTION_VIEW (\$09)

Ähnlich *BOX_OPTION* stellt auch dieser Befehl ein Optionsfeld auf dem Bildschirm dar. Allerdings kann diese Option vom Anwender nicht modifiziert werden.

Damit wird also nur der Inhalt des Optionswertes angezeigt (aktiv oder nicht aktiv), ein ändern des Wertes ist aber nicht möglich. Ein inaktives Optionsfeld wird zusätzlich auch in einer anderen Farbe dargestellt, der Wert für die Farbe findet sich in *C_InputFieldOff*.

BOX_STRING (\$0a)

BOX_STRING gibt einen Textstring auf dem Bildschirm aus. Dazu wird zuerst ein Eingabefeld auf dem Bildschirm dargestellt und dann der Text innerhalb des Feldes ausgegeben. Der Text kann vom Anwender geändert werden, indem das Eingabefeld mit der Maus angeklickt wird. Beendet wird die Eingabe mit **[RETURN]** oder mit einem Mausklick außerhalb des Eingabefeldes.

```

b BOX_STRING          ; Eingabefeld für Text
w TITEL               ; Zeiger auf Titel für Textfeld
w ROUTINE              ; Anwenderoutine
b RMOben +RTY1         ; y-Koordinate
w RMLinks +RTX1        ; x-Koordinate
w STRINGVEC            ; Zeiger auf Textstring
b 10                   ; Max. Zeichen im Eingabefeld

```

Wie bei *BOX_OPTION* dürften die Zeilen »TITEL«, »ROUTINE« und die Angabe der Koordinaten bereits bekannt sein.

Danach folgt ein Zeiger »STRINGVEC« (Word) auf einen Zwischenspeicher für die Texteingabe. Die Größe des Zwischenspeichers wird über das letzte Byte definiert.

Enthält der Zwischenspeicher bereits einen Text, dann wird dieser beim Aufbau des Registermenüs im Eingabefeld angezeigt. Wenn kein Text vorgegeben werden soll, dann muss das erste Byte im Zwischenspeicher ein \$00-Byte sein.

BOX_STRING_VIEW (\$0b)

Wie *BOX_STRING* stellt dieser Befehl ein Eingabefeld mit einem Textstring auf dem Bildschirm dar, der Text kann vom Anwender jedoch nicht verändert werden.

```

b BOX_STRING_VIEW      ; Anzeigefeld für Text
w TITEL                ; Zeiger auf Titel für Textfeld
w ROUTINE              ; Anwenderoutine
b RMOben +RTY1         ; y-Koordinate
w RMLinks +RTX1        ; x-Koordinate
w STRINGVEC            ; Zeiger auf Textstring
b 10                   ; Breite Eingabefeld

```

Die Breite des Eingabefeldes definiert hier nur die Größe des Feldes selbst, der String kann ggf. auch länger sein. Evtl. muss die Länge des Strings durch die Anwenderoutine über die Pixelbreite aller Zeichen gekürzt werden.

Ohne Begrenzung des rechten Randes für die Textausgabe über *rightMargin* kann Text, abhängig von der Breite aller Zeichen, über den rechten Rand hinaus angezeigt werden.

Um das zu vermeiden muss die Länge des Strings vor der Ausgabe über das Registermenü oder bei Verwendung der Routine *RegisterUpdate* auf die Länge in Pixel überprüft und eventuell gekürzt werden.

Im Gegensatz zu *BOX_OPTION_VIEW* wird das Eingabefeld für den Text aber nicht mit einer anderen Farbe dargestellt.

BOX_NUMERIC (\$0c)

Zur Eingabe von Zahlenwerten kann *BOX_NUMERIC* verwendet werden. Auch hier wird, wie bei *BOX_STRING*, ein Eingabefeld auf dem Bildschirm dargestellt.

Um das Aussehen der Zahl zu bestimmen, kann die Länge des Eingabefeldes mit verschiedenen Parametern verknüpft werden:

NUMERIC_LEFT	\$00	Zahl im Eingabefeld linksbündig ausgeben
NUMERIC_RIGHT	%1000 0000	Zahl im Eingabefeld rechtsbündig ausgeben
NUMERIC_SETSPC	\$00	Führende Leerstellen mit Leerzeichen füllen
NUMERIC_SET0	%0100 0000	Führende Leerstellen mit "0"-Zeichen füllen
NUMERIC_BYTE	\$00	Die Zahl ist ein Byte
NUMERIC_WORD	%0010 0000	Die Zahl ist ein Word (Low-/Highbyte)

Eine typischer Funktionsblock könnte wie folgt aussehen:

```

b BOX_NUMERIC          ; Eingabefeld für Zahlen
w BOX_TEXT             ; Zeiger auf Titel für Zahlenfeld
w BOX_ROUTINE          ; Anwenderoutine
b RMOben +RTY1         ; y-Koordinate
w RMLinks +RTX1        ; x-Koordinate
w DATAVEC             ; Zeiger Datenwert
b $04 ! NUMERIC_WORD ! NUMERIC_RIGHT

```

Wie bei den vorherigen Datentypen dürften die Zeilen »TITEL«, »ROUTINE« und die Angabe der Koordinaten bekannt sein.

»DATAVEC« zeigt auf eine Speicherstelle mit dem Datenwert, entweder als Byte oder Word. In diesem Beispiel kann der Datenwert max. vier Zeichen lang sein, was Zahlen im Bereich von 0 bis 9999 zulässt.

Eine Wertüberprüfung kann über die Anwenderoutine »BOX_ROUTINE« erfolgen. Ist der Wert ungültig, dann kann die Routine den eingegebenen Wert noch korrigieren, bevor die Registeroutine den Wert auf dem Bildschirm aktualisiert.

BOX_NUMERIC_VIEW (\$0d)

Ähnlich *BOX_NUMERIC* stellt diese Routinen Zahlen auf dem Bildschirm dar, jedoch ohne die Möglichkeit, diese durch den Anwender ändern zu lassen.

Die neuen Registermenü - Definitionen

Anbei nun eine Liste der zusätzlichen Definitionen, welche das Registermenü zur Verfügung stellt. Zuerst die neuen Konstanten:

.BOX_USER	= \$01	; Funktionsbereich definieren
.BOX_USER_VIEW	= \$02	; Wie BOX_USER aber nur anzeigen
.BOX_USEROPT	= \$03	; Optionsbereich definieren
.BOX_USEROPT_VIEW	= \$04	; Wie BOX_USEROPT aber nur anzeigen
.BOX_FRAME	= \$05	; Rahmen mit Titel zeichnen
.BOX_ICON	= \$06	; Icon zeichnen
.BOX_ICON_VIEW	= \$07	; Wie BOX_ICON aber nur anzeigen
.BOX_OPTION	= \$08	; Optionsfeld
.BOX_OPTION_VIEW	= \$09	; Wie BOX_OPTION aber nur anzeigen
.BOX_STRING	= \$0a	; Texteingabe
.BOX_STRING_VIEW	= \$0b	; Wie BOX_STRING aber nur anzeigen
.BOX_NUMERIC	= \$0c	; Zahleneingabe
.BOX_NUMERIC_VIEW	= \$0d	; Wie BOX_NUMERIC aber nur anzeigen
.NUMERIC_LEFT	= \$00	; Zahl linksbündig ausgeben
.NUMERIC_RIGHT	= \$80	; Zahl rechtsbündig ausgeben
.NUMERIC_SETSPC	= \$00	; Führende Leerzeichen ausgeben
.NUMERIC_SET0	= \$40	; Führende .0.-Zeichen ausgeben
.NUMERIC_BYTE	= \$00	; Zahlenwert als Byte
.NUMERIC_WORD	= \$20	; Zahlenwert als Word
.NO_OPT_UPDATE	= NULL	; keine Option aktualisieren
.USE_COLOR_INPUT	= \$ff	; Icon-Farbe = Eingabefeld
.USE_COLOR_REG	= \$ee	; Icon-Farbe = Registerfarbe

Hier die Sprungtabelle am Anfang der Registermenü-Routine.

.RegMenuBase	= \$6d00	; Ladeadresse Registermenü
.DoRegister	= \$6d00	; Registermenü starten
.ExitRegisterMenu	= \$6d03	; Registermenü beenden
.RegisterInitMenu	= \$6d06	; Registermenü initialisieren
.RegisterUpdate	= \$6d09	; Registerkarte aktualisieren
.RegisterAllOpt	= \$6d0c	; Alle Optionen aktualisieren
.RegisterNextOpt	= \$6d0f	; Nächste Option aktualisieren
.RegDrawOptFrame	= \$6d12	; Optionsrahmen zeichnen
.RegClrOptFrame	= \$6d15	; Optionsrahmen löschen
.RegisterSetFont	= \$6d18	; Font Registermenü aktivieren

Nach der Sprungtabelle folgt noch ein Byte, das Auskunft über die aktive Registerkarte gibt. Der Wert ist READ-ONLY, eine Änderung hat keine Auswirkung auf das Menü.

:regMenuCurrent	b \$00	; Bei \$6d1b
-----------------	--------	--------------

Die neuen Registermenü - Routinen

Teil B endet mit dem Kapitel 14 "Die restlichen Routinen".

Mit GEOS/MP3 gibt es komplett neue Routinen für das neue Registermenü, die im folgenden als ergänzendes Kapitel 15 "Das Registermenü" beschrieben werden.

15.1 DoRegister (\$6d00)

DoRegister installiert das Registermenü und zeichnet das Menü auf den Bildschirm. Soll zusätzlich zum Registermenü auch noch ein PullDown- oder Icon-Menü dargestellt werden, dann kann vor dem Aufruf von *DoRegister* noch die Routine *DoMenu* bzw. *Dolcons* aufgerufen werden.

Als einziger Parameter wird in *r0* die Adresse der Registermenü-Tabelle erwartet.

Da beim Aufbau des Registermenüs auch Anwender Routinen ausgeführt werden, können praktisch alle Register- und Systemadressen verändert werden.

Anwender Routinen dürfen das Register *r15* während dem Aufbau des Registermenüs nicht verändern, da hier ein Zeiger auf die aktuelle Registertabelle gespeichert wird.

15.2 ExitRegisterMenu (\$6d03)

Die Registermenü-Routine liegt im Bereich des Anwendungsspeichers. Bevor man wieder auf den gesamten Speicher zugreifen kann, sollte man *ExitRegisterMenu* aufrufen.

Damit werden Vektoren für den Zeichensatz und verschiedene GEOS-Vektoren wie *otherPressVec* zurückgesetzt. Wird der Bereich der Registermenü-Routinen überschrieben, dann kann ein Mausklick zu einem Systemabsturz führen, da die entsprechenden Routinen nicht mehr vorhanden sind.

Verändert werden Akku, x- und y-Register, sowie die das Register *r0*.

15.3 RegisterInitMenu (\$6d06)

Falls es notwendig wird, das aktuelle Register neu zu zeichnen, so verwendet man die Routine *RegisterInitMenu*.

Nach Aufruf der Routine wird das komplette Registermenü mit allen Kartenreitern und Menüeinträgen auf der aktuellen Registerseite neu gezeichnet. Das kann unter Umständen dann notwendig werden, wenn zwischenzeitlich der Bildschirm gelöscht wurde.

Verändert werden Akku, x- und y-Register, sowie die das Register *r0* bis *r15*.

15.4 RegisterUpdate (\$6d09)

RegisterUpdate aktualisiert einen einzelnen Registermenü-Eintrag. Die Routine kann dazu verwendet werden um innerhalb einer Anwender routine einen zusätzlichen Registermenü-Eintrag zu aktualisieren, z.B. wenn sich zwei Einträge innerhalb des Registermenü gegenseitig ausschließen.

Als Parameter wird in *r15* ein Zeiger auf den Registermenü-Eintrag benötigt.

Verändert werden Akku, x- und y-Register, sowie die das Register *r0* bis *r15*.

15.5 RegisterAllOpt (\$6d0c)

Löscht den Inhalt der aktuellen Registerkarte, zeichnet alle Kartenreiter und den Inhalt der aktiven Registerkarte neu. Es werden keine Parameter benötigt.

Da beim Aufbau des Registermenüs auch Anwenderroutinen ausgeführt werden, können praktisch alle Register- und Systemadressen verändert werden.

15.6 RegisterNextOpt (\$6d0f)

Die Routine *RegisterNextOpt* kommt immer dann zum Einsatz, wenn alle Inhalte der aktuellen Registerkarte neu gezeichnet werden sollen.

Es werden keine Parameter benötigt.

Da beim Aufbau des Registermenüs auch Anwenderroutinen ausgeführt werden, können praktisch alle Register- und Systemadressen verändert werden.

15.7 RegDrawOptFrame (\$6d12)

Die Routine *RegDrawOptFrame* wird nur intern vom Registermenü verwendet. Die Routine erwartet in den Registern *r2* bis *r4* die Koordinaten eines Rechtecks, das Format der Koordinaten entspricht der Routine *FrameRectangle*. Die Koordinaten werden von der Routine um jeweils 1 Pixel in alle Richtungen vergrößert.

In Abhängigkeit von *Flag_DrawFrame* wird der Rahmen um das Optionsfeld über *FrameRectangle* entweder gezeichnet (\$00) oder nicht gezeichnet (\$ff).

Nach dem Aufruf der Routine *RegDrawOptFrame* sind Akku, x- und y-Register, sowie die Adressen *r5* bis *r9* und *r11* verändert, die Koordinaten in *r3*, *r4*, *r2L* und *r2H* werden um 1 Pixel vergrößert.

15.8 RegClrOptFrame (\$6d15)

Wie *RegDrawOptFrame*, nur wird der Rahmen um ein Optionsfeld gelöscht.

15.9 RegisterSetFont (\$6d18)

Aktiviert den Registermenü-Zeichensatz. Es werden keine Parameter benötigt.

Die Routine *RegisterSetFont* verändert das Register *r0* und die Werte in den GEOS-Adressen *curHeight*, *baselineOffset*, *cardDataPntr*, *currentIndexTable*, *curSetWidth*.

2.8 Ausgelagerte Kernalroutinen

Die folgenden Routinen werden bei Bedarf vom GEOS/MegaPatch-Kernal in den Speicher eingelesen und dort ausgeführt. Nachdem die Routinen beendet wurden, wird der vorherige Speicherinhalt wieder hergestellt.

Eine Ausnahme bildet das Registermenü, welches im Anwendungsspeicher ausgeführt wird und damit den für Anwendungen zur Verfügung stehenden Arbeitsspeicher reduziert.

Hier eine Übersicht der ausgelagerten Kernalroutinen, die in der Systemspeicherbank von MP3 abgelegt sind. Die Adresse der Systemspeicherbank ist in *MP3_64_SYSTEM* abgelegt und ist die letzte, für den GEOS-DACC reservierte Speicherbank in der RAM-Erweiterung. Das muss bei einer Speichererweiterung größer als 4Mb nicht die letzte verfügbare Speicherbank sein.

Routine	Hinweise	Bereich	Größe	64K-Bank
Registermenü	1)	\$6D00 bis \$78ff	\$0c00	\$0000
EnterDeskTop: Desktop laden/starten		\$7E00 bis \$7fff	\$0200	\$0c00
Panic: Systemfehler anzeigen		\$8000 bis \$80ff	\$0100	\$0e00
ToBasic: Nach BASIC verlassen		\$8E00 bis \$8fff	\$0200	\$0f00
GetNxDay: Nächsten Tag berechnen		\$8000 bis \$807f	\$0080	\$1100
DoAlarm: Alarm auslösen/anzeigen		\$8000 bis \$807f	\$0080	\$1180
Dialogbox: Erweiterte GetFiles-Routine	2)	\$6000 bis \$7bff	\$1c00	\$1200
GetFiles-Routine: Daten einlesen	3)	\$851f bis \$869e	\$0180	\$2e00
GetFiles-Routine: Dialogbox-Menü		\$8000 bis \$837f	\$0380	\$2f80
Dialogbox: Hintergrund laden/speichern		\$8000 bis \$82ff	\$0300	\$3300
Dialogbox: Hintergrund/Farbdaten			\$03e8	\$3600
Dialogbox: Hintergrund/Grafikdaten			\$1f40	\$39e8
GetBackScreen: Hintergrundbild anzeigen		\$8000 bis \$80ff	\$0100	\$5928
Hintergrundbild: Farbdaten			\$03e8	\$5a28
Hintergrundbild: Grafikdaten			\$1f40	\$5e10
ScreenSaver: Bildschirmschoner anzeigen	1) 2)	\$6400 bis \$7fff	\$1c00	\$7d50

Routine	Hinweise	Bereich	Größe	64K-Bank
Bildschirmschoner: Hintergrund/Farbdaten			\$03e8	\$9950
Bildschirmschoner: Hintergrund/Grafikdaten			\$1f40	\$9d38
Druckerspooler: Menü und Druckroutine	2)	\$4000 bis \$55ff	\$1600	\$bc78
Druckerspooler: Infoblock Druckertreiber	5)		\$0100	\$d180
Druckerspooler: Druckertreiber	4)	\$7900 bis \$7f3f	\$0640	\$d280
Druckertreiber: Infoblock			\$0100	\$d8c0
Druckertreiber: Für Anwendungen	4)	\$7900 bis \$7f3f	\$0640	\$d9c0
TaskManager: Hauptmenü und Manager	2)	\$4000 bis \$5fff	\$2000	\$e000

Hinweise

1. Diese Routinen können auch vom Anwender aufgerufen werden. Man sollte jedoch darauf achten, das eigene Programme die Startadresse der Routine im *APP_RAM* nicht überschreiben.

Wenn die Routinen nicht benötigt werden, dann kann der komplette Anwendungsspeicher von \$0400 bis \$7fff für Anwendungen genutzt werden.

2. Diese Routinen werden aus dem Kernal heraus aufgerufen und werden im Anwendungsspeicher ausgeführt. Während die Routinen ausgeführt werden, wird der Speicherinhalt in der Speichererweiterung ausgelagert. Eigene Programme müssen deshalb keinerlei Rücksicht auf diese Routinen und deren Lage im Anwendungsspeicher nehmen.
3. Die Routinen werden nicht im normalen Anwendungsspeicher ausgeführt, da diese Daten aus der Anwendung benötigen und diese überall im Anwendungsspeicher abgelegt sein könnten.

Als Startadresse wurde daher *dlgBoxRamBuf* gewählt, der Bereich ist im Normalfall dem GEOS-Kernal vorbehalten.

4. Auf Grund eines Fehlers im Programm "GeoCalc64" wurde in den MP3-Versionen ab 2018 eine zusätzliche Option integriert, um die Größe von "Druckertreiber im RAM" und "Druckerspooler/Druckertreiber" um jeweils 1 Byte zu reduzieren. Ohne diese Option kann "GeoCalc64" bei der Verwendung bestimmter Druckertreiber abstürzen.

Die Option findet sich im "GEOS.Editor" unter Drucker/GeoCalc-Fix.

5. Reserviert für den Infoblock des Druckerspooler/Druckertreiber.

Der Bereich ist für künftige Erweiterungen in MP3 reserviert, da beim laden des Druckertreibers über eine Anwendung der Infoblock des eigentlichen Druckertreibers nach *fileHeader* geladen wird.

2.8.1 Sprungtabelle für ausgelagerte Kernalroutinen

Diese Routinen setzen die Register *r0* bis *r3L* auf die externen Kernal-Funktionen. Danach kann mittels *StashRAM*, *FetchRAM* oder *SwapRAM* auf diese Funktionen zugegriffen werden. Wenn die erweiterten Kernalroutinen angepasst werden sollen, dann darf die Größe den im Register *r2* enthaltenen Wert nicht überschreiten.

Adresse	Routine	Beschreibung
\$cfed	SetADDR_TaskMan	*) TaskManager-Menü.
\$cfe6	SetADDR_Register	Registermenü-Routine.
\$cfe3	SetADDR_EnterDT	EnterDeskTop-Routine.
\$cfe0	SetADDR_ToBASIC	*) Routine zum beenden von GEOS.
\$cfdd	SetADDR_PANIC	*) PANIC!-Dialogbox.
\$cfda	SetADDR_GetNxDay	*) Datumswechsel.
\$cfd7	SetADDR_DoAlarm	*) Die Alarmroutine.
\$cfd4	SetADDR_GetFiles	*) Dateiauswahlbox.
\$cfd1	SetADDR_GFilData	*) GetFiles: Einlesen der Tabelleneinträge.
\$cfce	SetADDR_GFilMenu	*) GetFiles: Zeichnen Auswahlbox-Menüs.
\$cfcb	SetADDR_DB_SCRN	*) Wird eine Dialogbox geöffnet so rettet MegaPatch den Inhalt des Bildschirms in einen geschützten Bereich in der Speichererweiterung. Nach beenden der Dialogbox kopiert diese Routine den Bildschirminhalt wieder in den Vordergrundbildschirm.
\$cfc8	SetADDR_DB_GRFx	*) Grafikdaten unter Dialogbox.
\$cfc5	SetADDR_DB_COLS	*) Farbdaten unter Dialogbox.
\$cfc2	SetADDR_BackScrn	*) Diese Routine kopiert das festgelegte Hintergrundbild in den Vordergrundbildschirm. Programmierer müssen diese Routine nicht selbst ausführen, denn dafür existiert ein Einsprung in der Sprungtabelle bei GetBackScreen (\$c0b8).
\$cfbf	SetADDR_ScrSaver	Der aktuelle Bildschirmschoner.
\$cfbc	SetADDR_Spooler	*) Das Druckerspooler-Menü.
\$cfb9	SetADDR_PrnSpool	*) Zeigt auf Druckertreiber für Spooler.
\$cfb6	SetADDR_PrnSpHdr	*) Der Infoblock des Druckerspools. Dieser Bereich ist für künftige Erweiterungen reserviert, da der Druckerspooler kein eigenständiger Druckertreiber ist. Wird der Druckerspooler eingelesen (z.B. wenn GeoWrite gestartet wird) dann lädt GetFile den Infoblock des aktuellen Druckertreibers.
\$cfb3	SetADDR_Printer	Der aktuelle Druckertreiber. Dieser muss sich nicht mehr auf jeder Diskette befinden sondern wird nun wie beim C128 ständig im Speicher gehalten.
\$cfb0	SetADDR_PrntHdr	Der Infoblock zum aktuellen Druckertreiber.

*) Die Routinen werden nur intern vom GEOS/MegaPatch-Kernal verwendet und sind nicht für den Einsatz in eigenen Programme vorgesehen.

2.8.2 EnterDeskTop (\$c22c)

Die Routine *EnterDeskTop* versucht den aktuellen DeskTop zu starten.

Dabei sucht der MP3-Kernal zuerst auf RAM-Laufwerken nach einer Kopie des DeskTop. Wird hier kein DeskTop gefunden, dann wird die Suche auf Diskettenlaufwerken fortgesetzt. Wenn auch hier kein DeskTop gefunden wird, so erscheint die bekannte Fehlermeldung, das man eine Diskette mit DeskTop einlegen soll:



Bild 2.4: Bitte Diskette mit deskTop einlegen

Die dazugehörige Dialogbox befindet sich auch weiterhin im Kernal, da einige DeskTop-Programme diese verändern.

Um ein Höchstmaß an Kompatibilität zu erreichen, sollte innerhalb des GEOS-Kernal nichts verändert werden. Auch der Name der DeskTop-Umgebung sollte hier nicht angepasst werden, da nicht sichergestellt ist, dass spätere GEOS-Versionen den Namen an der gleichen Stelle im Speicher erwarten.

Einfacher ist es die eigenen Desktop-Oberfläche in "DESK TOP" umzubenennen.

Es gibt aber die Möglichkeit eine Desktop-Oberfläche zu starten, deren Name nicht im Kernal hinterlegt ist. Auch TopDesk aus der ursprünglichen MP3-Version aus dem Jahr 2000 verwendet diese Möglichkeit.

Dabei wird die ausgelagerte Kernalroutine *EnterDeskTop* durch eine eigene Routine ersetzt. Die eigene Routine kann dann jede Art von DeskTop laden.

Damit später eine Rückkehr zum originalen DeskTop möglich ist, sollte die Original-Kernalroutine vor der Änderung zwischengespeichert und vor der Rückkehr zum DeskTop wieder hergestellt werden.

Die Original-Kernalroutine kann dann in einer freien 64K-Speicherbank innerhalb der Speichererweiterung abgelegt werden.

Hier ein Beispiel für eine angepasste EnterDeskTop-Routine:

```
:InstallNewDT    jsr    FindFreeBank    ; Freie Speicherbank suchen
                  txa                      ; Gefunden?
                  bne    :error          ; Nein, Abbruch...

                  jsr    AllocateBank    ; Speicherbank reservieren
```

```

        jsr    SetADDR_EnterDT    ; Original EnterDeskTop-Routine
        jsr    FetchRAM           ; in Computer-Speicher einlesen

        lda    #$00              ; Zeiger auf Anfang der freien
        sta    r1L               ; Speicherbank setzen.
        sta    r1H
        lda    FREE_BANK         ; Adresse Speicherbank setzen
        sta    r3L
        jsr    StashRAM          ; Original-Routine speichern

        ...                     ; Ggf. DeskTop laden und ebenfalls
                                ; in der Speicherbank ablegen

```

Zu Beginn wird eine freie 64K-Speicherbank gesucht. Die Routine speichert die Adresse der gefundenen 64K-Speicherbank ab dem Label »:FREE_BANK«.

Danach wird die Original-Routine zu *EnterDeskTop* über *FetchRAM* eingelesen. Die *EnterDeskTop*-Routine liegt dann von \$7e00 bis \$7fff im *APP_RAM*.

Danach setzt man die Zieladresse für *StashRAM*. *r1* zeigt auf den Bereich in der freien Speicherbank und *r3L* auf die Speicherbank selbst. Mit *StashRAM* wird die Original-Routine dann in der freien Speicherbank gesichert.

Jetzt kann die Routine *EnterDeskTop* durch eine eigene Routine ersetzt werden.

```

:InstOwnDT    jsr    SetADDR_EnterDT    ; Zeiger auf EnterDeskTop-Routine

              LoadW  r0,OwnLoader       ; Zeiger auf eigene Routine
              jsr    StashRAM           ; Neue Routine für EnterDeskTop
                                      ; in REU speichern

```

Die neue DeskTop-Laderoutine befindet sich nun im erweiterten GEOS-Speicher und wird immer dann aufgerufen, wenn eine Applikation beendet wurde. Eine solche RAM-Startroutine könnte wie folgt aussehen:

```

;GEOS initialisieren.
:OwnLoader    sei                ; Interrupt sperren
              cld
              ldx    #$ff        ; Wichtig für AutoBoot-Vorgang
              stx    firstBoot
              txs

; Variablen und Bildschirm löschen.
              jsr    GEOS_InitSystem ; GEOS-Werte zurücksetzen
              jsr    ResetScreen    ; Bildschirm löschen

:FetchOwnDT   LoadW  r0,APP_RAM     ; APP_RAM = $0400
              LoadW  r1,$0200      ; Startadresse DeskTop im DAC
              LoadW  r2,$4000      ; Max. DeskTop-Größe
:OwnDTBank    lda    $xx           ; RAM-Bank in der sich der
              sta    r3L           ; eigene DeskTop befindet
              jsr    FetchRAM      ; DeskTop einlesen.

              LoadB  r0L,%00000000 ; Kein Datenfile nachladen
              LoadW  r7,APP_RAM     ; Anwendung ab APP_RAM starten
              jmp    StartAppl

```

Ab dem Label »:OwnLoader« findet die Standard-Initialisierung für GEOS, diese sollte in jeder DeskTop-Routine enthalten sein. *GEOS_InitSystem* und *ResetScreen* sind Teil der erweiterten Sprungtabelle in MP3. Damit sollten keine Anpassungen an spezielle MP3-Versionen erforderlich werden.

Ab »:FetchOwnDT« steht die Routine, welche dann den eigenen DeskTop startet. In diesem Beispiel wurde der eigene DeskTop zuvor in einer zusätzlichen Speicherbank abgelegt und über *FetchRAM* eingelesen und über *StartAppl* gestartet.

Die Routine »:FetchOwnDT« könnte auch *FindFTypes/FindFile* verwenden, um eine DeskTop-Datei auf Diskette zu suchen.

Die Routine ab »:OwnLoader« sollte im Speicher frei verschiebbar sein, also intern keine absoluten Adressen verwenden. Man sollte sich nicht auf eine bestimmte Adresse festlegen, da spätere MP3-Versionen die Routine evtl. in einem anderen Speicherbereich ausführen können.

2.8.3 ToBasic (\$c241)

Diese Routine ist aus GEOS V2.x übernommen worden, wurde aber geringfügig optimiert z.B. wird darauf gewartet, das kein Feuerknopf gedrückt ist.



ACHTUNG! Wer eine CMD-RAMLink verwendet und dort ein AutoStart-Menü eingerichtet hat, der kann keine Programme unter GEOS starten. Nach einem Reset wird automatisch das RAMLink-AutoBoot-Menü aktiviert.

2.8.4 Panic (\$c2c2)

Jeder Programmierer kennt die Panic!-Dialogbox, mancher Anwender leider auch. Diese Routine erscheint wenn innerhalb einer Anwendung, wenn ein ungültiger Assembler-Befehl *brk* auftaucht ("Systemfehler nahe \$....").

Die erweiterte Panic!-Dialogbox fängt leichte Fehler über den *BRKVector* ab und kehrt danach zum DeskTop zurück. Bei schweren Fehlern (illegale Opcodes) hilft auch diese Routine nicht mehr weiter und GEOS muss dann neu gestartet werden.

2.8.5 GetNxDay (intern)

Diese Routine ist relativ uninteressant für den Programmierer. Sie wird alle 24 Stunden vom GEOS-Kernal aufgerufen, wenn in der GEOS-Uhr ein Überlauf stattfindet. Dann wird mit dieser Routine das Datum auf den nächsten Tag gesetzt.

2.8.6 DoAlarm (intern)

Wird mit dem GEOS-Wecker eine Alarmzeit eingestellt, dann wird vom GEOS-Kernal nach Erreichen der Alarmzeit diese Routine ausgeführt. Im Gegensatz zur Originalroutine zeigt der MP3-Kernal den Alarm akustisch und optisch an.

2.8.7 GetFiles (intern)

Die neue Dateiauswahlbox von GEOS/MegaPatch. Diese Routine wird im Speicher ab \$6000 ausgeführt. Beispiel für eine Dateiauswahlbox:



Bild 2.5: Dateiauswahlbox

Der Screenshot zeigt die Dateiauswahlbox unter GeoWrite. Im Gegensatz zur alten Dateiauswahlbox werden hier bis zu 255 Dateien in der Dateiliste angezeigt. Mit Hilfe der Scrollpfeile und des Scrollbalkens kann man innerhalb der Liste navigieren.

Die Dateiauswahlbox enthält außerdem die vier Laufwerk-Icons des MP3-Systems. Diese können mit den entsprechenden Patches dazu verwendet werden das Laufwerk zu wechseln. Siehe dazu auch **Teil D, Kapitel 3.1 ab Seite 480** im Abschnitt zu *DoDlgBox* und *DBSETDRVICON*.

Die notwendigen Patches für GeoWrite (und evtl. auch für andere Anwendungen) sind nicht Teil von GEOS/MegaPatch.

Wer eine bestimmte Datei sucht, der kann im Feld "Eintrag suchen" einzelne Zeichen eingeben. Nach der Eingabe von **[RETURN]** sucht die Auswahlbox nach der ersten Datei, die mit den bisher eingegebenen Zeichen beginnt.

Jeder weitere Druck auf **[RETURN]** zeigt die nächste Datei an. Findet die Auswahlbox einen Eintrag der genau der eingegebenen Zeichenfolge entspricht, dann wird die Datei automatisch geöffnet. Das Eingabefeld beachtet dabei die Groß/Kleinschreibung!

Unterhalb des "Eintrag suchen"-Feldes wird der Name der aktuell ausgewählten Datei angezeigt. Außerdem werden noch Datum und Uhrzeit der letzten Änderung für die Datei, die aktuelle Dateigröße und der Schreibschutz angezeigt. Der Schreibschutz kann mit einem Mausklick geändert werden.

Am unteren Rand der Dateiauswahlbox findet sich die Option "Dateien sortieren". Mit einem Mausklick wird die Liste der Dateien alphabetisch sortiert.

Dies funktioniert nicht wenn die Dateiauswahlbox über den Funktionscode *DBUSRFILES* aufgerufen wird: Hier könnte die Dateitabelle mit anderen Tabellen verknüpft sein, was im schlimmsten Fall zur falschen Auswertung eines Eintrages führen könnte.

2.8.8 GetFiles_Data (intern)

Die interne Routine wird von *GetFiles* aufgerufen und überträgt die benötigten Einträge für die Auswahlbox in die Speichererweiterung. Bei *DBGETFILES* werden dazu die Dateien über *FindFTypes* eingelesen und bei *DBUSRFILES* aus dem vom Anwender bestimmten Speicherbereich für die Auswahlbox.

Da bei *DBUSRFILES* praktisch der komplette Speicher im *APP_RAM* als Ablagebereich für Tabelleneinträge dienen kann, wird diese Routine im Bereich von *dlgBoxRamBuf* ausgeführt. Der Bereich ist normalerweise dem Kernal vorbehalten.

Für den Anwender hat diese Routine keinerlei Bedeutung.

2.8.9 GetFiles_Menu (intern)

Auch diese Routine wird nur intern von *GetFile* aufgerufen und zeichnet die Auswahlbox mit allen Icons auf den Bildschirm.

Die Routine wurde ausgelagert da es sonst zu Problemen bei der Initialisierung der Dialogbox-Icons mit *Dolcons* kommen kann. Würde die Dateiauswahlbox aus einem Programm heraus gestartet welches ab \$6000 im Speicher liegt, dann könnte *Dolcons* die selbstdefinierten Icons nicht korrekt darstellen, da der Bereich jetzt von der neuen Auswahlbox überschrieben wurde.

Für den Anwender hat diese Routine keinerlei Bedeutung.

2.8.10 TaskManager (intern)

Der TaskManager erlaubt es bis zu neun Anwendungen gleichzeitig geöffnet zu halten. Mittels der Tastenkombination **[CBM] + [CTRL]** wird der TaskManager aktiviert. Dies funktioniert nur dann wenn sich GEOS gerade in der MainLoop befindet, also keine andere Anwendung den Prozessor für sich reserviert.

Der TaskManager wird unter MP3 nach \$4000 in den Speicher eingelesen und gleichzeitig wird der Anwendungsspeicher in die Speichererweiterung ausgelagert.

Die ersten drei Byte beinhalten einen Sprungbefehl zum Hauptprogramm. Danach folgt eine Datentabelle:

:TaskBank00	b \$00
:TaskBank01	b \$00
:TaskBank02	b \$00
:TaskBank03	b \$00
:TaskBank04	b \$00
:TaskBank05	b \$00
:TaskBank06	b \$00
:TaskBank07	b \$00
:TaskBank08	b \$00

Diese Bytes enthalten für jeden installierten Task die reservierten 64K-Speicherbank, eine Bank für jeden Task. Nicht installierte Tasks enthalten hier den Wert \$00.

Unter MegaPatch128 wird in diesen Speicherbänken der Inhalt von Bank 1 (FrontRAM) des Computer-RAM abgelegt. Für den Inhalt des VDC und Bank 0 (BackRAM) werden jeweils weitere 64K benötigt. Somit benötigt ein Task immer 128K.

Für MegaPatch64 sind 64K für einen Task ausreichend.

Danach folgt eine Tabelle, der Inhalt darüber Auskunft gibt, ob ein Task in Verwendung ist oder nicht:

:BankInUse00	b \$00
:BankInUse01	b \$00
:BankInUse02	b \$00
:BankInUse03	b \$00
:BankInUse04	b \$00
:BankInUse05	b \$00
:BankInUse06	b \$00
:BankInUse07	b \$00
:BankInUse08	b \$00

Der Wert \$ff zeigt an, dass der entsprechende Task durch eine geöffnete Anwendung belegt ist. Wird eine Applikation durch den Befehl *jmp EnterDeskTop* beendet, dann wird hier der Inhalt der entsprechenden Variable gelöscht und der Task steht wieder für neue Anwendungen zur Verfügung.

Es folgt ein Byte, das die Anzahl der installierten Tasks definiert.

:MaxTaskInstalled	b \$09
-------------------	--------

Die MP3-Demo-Version enthielt hier den Wert \$02 (maximale Anzahl Tasks war auf zwei Anwendungen beschränkt). Seit 2018 gibt es diese Beschränkung nicht mehr.

Unter MegaPatch128 folgen hier zwei weitere Tabellen, die zusätzliche Speicherbänke für den Inhalt des VDC und der Bank 0 definieren:

:TaskVDCBank00	b \$00
:TaskVDCBank01	b \$00
:TaskVDCBank02	b \$00
:TaskVDCBank03	b \$00
:TaskVDCBank04	b \$00
:TaskVDCBank05	b \$00
:TaskVDCBank06	b \$00
:TaskVDCBank07	b \$00
:TaskVDCBank08	b \$00
:Task0Bank00	b \$00
:Task0Bank01	b \$00
:Task0Bank02	b \$00
:Task0Bank03	b \$00
:Task0Bank04	b \$00
:Task0Bank05	b \$00
:Task0Bank06	b \$00
:Task0Bank07	b \$00
:Task0Bank08	b \$00

Für jeden Task wird auch ein Name festgelegt, der entweder aus dem Namen der Anwendung oder dem Namen des Dokuments gebildet wird.

Da unter GEOS/MegaPatch128 zusätzliche Daten benötigt werden, ist die Adresse der Namen unter GEOS64 und GEOS128 unterschiedlich:

- MegaPatch64 = \$4016
- MegaPatch128 = \$4028

Für jeden Task sind 16 Zeichen + *NULL*-Byte reserviert:

```
:TaskNam00      s 17
:TaskNam01      s 17
:TaskNam02      s 17
:TaskNam03      s 17
:TaskNam04      s 17
:TaskNam05      s 17
:TaskNam06      s 17
:TaskNam07      s 17
:TaskNam08      s 17
```

Für jeden Task findet man hier den Namen der Anwendung oder des Dokuments, mit dem der neue Task geöffnet wurde.

Wird innerhalb der laufenden Anwendung eine andere Anwendung oder ein anderes Dokument geöffnet, dann bleibt hier der ursprüngliche Taskname erhalten.

2.8.11 DB_SCREEN (intern)

Die neue Dialogbox von MP3 rettet den Bildschirminhalt vor Aufbau der Dialogbox in einen reservierten Bereich in der Speichererweiterung. Nach Abbau der Dialogbox wird der Bildschirminhalt wieder hergestellt.

Diese Routine übernimmt beide Aufgaben:

Der Einsprung bei *DB_SCREEN_SAVE* speichert den aktuellen Bildschirminhalt während *DB_SCREEN_LOAD* den Bildschirminhalt wiederherstellt.

```
; Bildschirm-Inhalt speichern
      jsr   SetADDR_DB_SCRN ; Zeiger auf DB_SCREEN setzenden
      jsr   SwapRAM        ; Routine nach $8000 einlesen

      jsr   DB_SCREEN_SAVE ; Bei $8000

; Bildschirm-Inhalt zurücksetzen.
      jsr   SetADDR_DB_SCRN ; Zeiger auf DB_SCREEN setzenden
      jsr   SwapRAM        ; Routine nach $8000 einlesen

      jsr   DB_SCREEN_LOAD ; Bei $8003
```

Für den Anwender hat diese Routine keinerlei Bedeutung, da die Verwendung in eigenen Programmen auf Grund der benötigten Registerinhalte nicht möglich ist.

2.8.12 GetBackScreen (\$c0e8)

Diese Routine lädt das Hintergrundbild aus dem erweiterten Speicher und stellt es im Vordergrundbildschirm dar. Ist kein Hintergrundbild definiert, dann wird der Bildschirm mit dem Füllmuster aus *BackScrPattern* gefüllt.

Für diese Routine existiert ein Einsprung innerhalb der neuen Sprungtabelle. Siehe dazu **Teil D, Kapitel 4.34 ab Seite 487**.

2.8.13 Bildschirmschoner (intern)

Der Bildschirmschoner wird vom Kernal automatisch aktiviert, wenn für eine festgelegte Zeit keine Maustaste gedrückt wurde und keine Tastatureingaben erfolgt sind. Der Bildschirmschoner wird ab \$6400 in den Computerspeicher geladen.

Es folgt der interne Aufbau der Initialisierungsroutine:

```

; Bildschirmschoner starten.
:MainInit      jmp      InitScrSvr

; Bildschirmschoner initialisieren.
:InstallSvr    jmp      InstScrSvr      ; Bei bedarf Speicher reservieren

:SaverName     b "Starfield" ,NULL

:InitScrSvr    php
               sei

               ldx      #(r15H - r0L)    ; ZeroPage speichern
::51           lda      r0,x
               pha
               dex
               bpl      :51

               jsr      DoSaverJob      ; Effekt starten

               lda      #%01000000      ; ScreenSaver-Reset
               sta      Flag_ScrSaver

               ldx      #0              ; ZeroPage laden
::52           pla
               sta      r0,x
               inx
               cpx      #(r15H - r0L) +1
               bne      :52

               plp
               rts

```

Diese Routine sollte in jedem Bildschirmschoner enthalten sein:

Wichtig ist, das die ersten Byte einen Sprungbefehl auf den Bildschirmschoner und einen Sprungbefehl auf die Installationsroutine beinhalten. Wird keine Installation benötigt, kann der Sprungbefehl durch **ldx #\$00; rts** ersetzt werden.

Für den Fall, das der Bildschirmschoner bei der Installation Teile nachladen muss, ist das Laufwerk, von welchem der Bildschirmschoner geladen wurde, noch aktiv.

Außerdem muss der Inhalt des Bildschirms abgespeichert werden. Das gleiche gilt für Register und Speicherbereiche welche der Bildschirmschoner verändert.

Wichtig ist auch das die Register *r0* bis *r15* unverändert bleiben, was durch die obige Routine gegeben ist.

Zum speichern des Bildschirms kann man folgende Routinen verwenden:

```
:SaveScreen    LoadW  r0, SCREEN_BASE
                LoadW  r1, R2_ADDR_SS_GRAFX
                LoadW  r2, R2_SIZE_SS_GRAFX

                lda     MP3_64K_SYSTEM
                sta     r3L

                jsr     StashRAM

                LoadW  r0, COLOR_MATRIX
                LoadW  r1, R2_ADDR_SS_COLOR
                LoadW  r2, R2_SIZE_SS_COLOR

;              lda     MP3_64K_SYSTEM    ; Bereits gesetzt
;              sta     r3L

                jsr     StashRAM
```

Bevor man den Bildschirmschoner beendet, muss der Bildschirminhalt wieder zurückgesetzt werden. Dazu kann man die folgende Routine verwenden:

```
:LoadScreen    LoadW  r0, SCREEN_BASE
                LoadW  r1, R2_ADDR_SS_GRAFX
                LoadW  r2, R2_SIZE_SS_GRAFX

                lda     MP3_64K_SYSTEM
                sta     r3L

                jsr     FetchRAM

                LoadW  r0, COLOR_MATRIX
                LoadW  r1, R2_ADDR_SS_COLOR
                LoadW  r2, R2_SIZE_SS_COLOR

;              lda     MP3_64K_SYSTEM    ; Bereits gesetzt
;              sta     r3L

                jsr     FetchRAM
```

Für den Bildschirm und die Farbdaten sind spezielle Bereiche reserviert. Die Größe des Bildschirmschoners ist auf \$1c00 Bytes beschränkt, was aber ausreichen dürfte.

Die Abfrage von Tastatur und Maus könnte wie folgt aussehen:

```
                lda     #$00
                sta     $dc00
                lda     $dc01
                eor     #$ff
                bne     EndScreenSaver
```

Man sollte auf eine Abfrage der Mausbewegung verzichten, da bei installierter CMD-SuperCPU oder eines TurboChameleon64 auf Grund der schnelleren Abfrage der Mausrichtungsregister die Maus leicht "zittert", was zur sofortigen Beendigung des Bildschirmschoners führen würde.

2.9 Die Laufwerkstreiber für NativeMode

Die Laufwerkstreiber wurden in MP3 ebenfalls überarbeitet. Was nicht verändert werden konnte war das GEOS-TurboDOS, da hier bereits kleinste Änderungen zu Timing-Problemen führen und das System instabil wird. Auch innerhalb der Treiber mussten bestimmte Routinen unverändert bleiben, da sonst bei einigen GEOS-Anwendungen der Kopierschutz nicht mehr funktioniert (GeoPublish, GeoWrite...).

2.9.1 Das Diskettenformat NativeMode

Neben den Diskettenformaten 1541, 1571 und 1581 wurde mit erscheinen der CMD-Hardware (Festplatte CMD-HD, 3,5"-Diskette CMD-FD und Speicher CMD-RAMLink) ein neues Diskettenformat für GEOS eingeführt: Der NativeMode.

Das Diskettenformat verwaltet bis zu 255 Spuren zu je 256 Sektoren, BAM und Verzeichnis liegen ab Track \$01 und es werden Unterverzeichnisse unterstützt.

Hier der Aufbau einer NativeMode-Diskette:

Diskettengröße		
Spuren:	Sektoren:	Gesamt:
1 bis 255	0 bis 255	255 x 256 = 65280 Blocks

ROOT- und SUBDIR-Verzeichnisheader

Für das Hauptverzeichnis (ROOT) findet sich der Block in Sektor \$01,\$01, für Unterverzeichnisse (SUBDIR) ist die Block-Adresse unterschiedlich.

Byte	Hex.	Beschreibung
0	\$00	Track des ersten Directory-Block (ROOT = \$01)
1	\$01	Sektor des ersten Directory-Block (ROOT = \$22)
2	\$02	ASCII "H" (\$48) als Formatkennung
3	\$03	Reserviert (\$00)
4 - 21	\$04 - \$15	Disketten-Name, aufgefüllt mit \$a0-Byte (Shift-Space)
22 - 23	\$16 - \$17	Disketten-ID
24	\$18	Shift-Space (\$a0) als Trennung
25	\$19	ASCII "1" (\$31) für die DOS-Version
26	\$1a	ASCII "H" (\$48) für die Formatkennung
27 - 28	\$1b - \$1c	Shift-Space (\$a0) als Trennung
29 - 31	\$1d - \$1f	Reserviert (\$00)
32	\$20	Track des ROOT-Header (\$01)
33	\$21	Sektor des ROOT-Header (\$01)
34	\$22	Track des SUBDIR-Header
35	\$23	Sektor des SUBDIR-Header
36	\$24	Track des Directory-Block im Elternverzeichnis für SUBDIR
37	\$25	Sektor des Directory-Block im Elternverzeichnis für SUBDIR
38	\$26	Zeiger auf Byte innerhalb des Directory-Block
39 - 255	\$27 - \$ff	NULL-Byte, reserviert

NativeMode BAM (Erster BAM-Block)

Der erste BAM-Block findet sich auf allen NativeMode-Disketten in Block \$01,\$02.

Byte	Hex.	Beschreibung
0	\$00	Reserviert
1	\$01	Reserviert
2	\$02	ASCII "H" (\$48) für die Formatkennung
3	\$03	Verifizierung für Formatkennung (\$b7 = \$48 eor \$ff)
4 - 5	\$04 - \$05	Disketten-ID
6	\$06	Nicht verwendet, enthält der Vorgabewert einer 1581 (\$c0)
7	\$07	Flag für "Auto Loader" Datei (\$00)
8	\$08	Anzahl Tracks (1-255) auf Diskette
9 - 31	\$09 - \$1f	Reserviert (\$00)
32 - 255	\$20 - \$ff	BAM für Spur 1 bis 7 (32 Byte für jede Spur, immer \$00)

NativeMode BAM (Block 3 bis 33)

Die restlichen Blöcke der BAM haben das folgende Format:

Spur	Sektor	Beschreibung
1	3	BAM für Spur 8 bis 15 (32 Byte für jede Spur)
1	4	BAM für Spur 16 bis 23 (32 Byte für jede Spur)
1	5	BAM für Spur 24 bis 31 (32 Byte für jede Spur)
...		
1	31	BAM für Spur 232 bis 239 (32 Byte für jede Spur)
1	32	BAM für Spur 240 bis 247 (32 Byte für jede Spur)
1	33	BAM für Spur 248 bis 255 (32 Byte für jede Spur)

NativeMode BAM - BAM-Format

Byte 32 bis 255 in Block \$01,\$02 und Byte 0 bis 255 in Block \$01,\$03 bis \$21.

Byte	Beschreibung
0	BAM-Status für Sektor 0-7
1	BAM-Status für Sektor 8-15
2	BAM-Status für Sektor 16-23
...	
29	BAM-Status für Sektor 232-239
30	BAM-Status für Sektor 249-247
31	BAM-Status für Sektor 248-255

Hinweis: Das unterste Bit (b0) im Byte definiert den Status des höchsten Sektor, der durch das Byte abgedeckt wird. Ein Wert von 1 zeigt an, das der Block verfügbar ist, ein Wert von 0 zeigt an das der Block belegt ist.

2.9.2 NativeMode - Bootsektor

Sektor \$01,\$00 auf einer NativeMode-Partition ist für einen Bootsektor reserviert, der Aufbau entspricht dem einer 1571-Diskette. Damit kann der C128 automatisch von einer NativeMode-Partition booten.

Der Bootsektor wird vom Kernal nach \$0b00 in Bank 0 geladen und ausgewertet:

Offset	Typ	Beschreibung
\$00	String	Kennung für den Bootsektor "CBM" = \$43,\$42,\$4d.
\$03	Word	Ladeadresse für bis zu 255 zusätzliche Sektoren ab Track 1/Sektor 1.
\$05	Byte	Bank-Adresse im C128-RAM (Bank 0/1).
\$06	Byte	Anzahl zusätzliche Sektoren (0-255).
\$07	String	Name für die Meldung "BOOTING ...". Ende mit <i>NULL</i> -Byte. Wenn kein Text ausgegeben werden soll, dann steht hier <i>NULL</i> .
\$xx	String	Name des zu bootenden Programms. Ende mit <i>NULL</i> -Byte. Wenn kein Programm gestartet werden soll, dann steht hier <i>NULL</i> .
\$yy	Programm	Nach den beiden Strings kann ein Programm folgen, das vom C128 ausgeführt wird, wenn kein Programm automatisch geladen und gestartet werden soll.

Wenn keine zusätzlichen Sektoren nachgeladen werden sollen, dann sind die Bytes ab Offset \$03 bis \$06 gleich \$00.

Der Bootsektor kann auch als Datei "bootsect.128" innerhalb eines Verzeichnisses auf der SD-Karte in einem SD2IEC gespeichert werden. Dabei werden von der SD2IEC-Firmware aktuell die Werte \$03 bis \$06 ignoriert, man kann also nur ein Programm innerhalb des Bootsektors ausführen oder ein Programm automatisch laden und starten. Die Datei ist dann max. 256 Byte groß.

Hier ein Beispiel für einen Bootsektor:

```
o $0b00

b "CBM"                ; Boot-Kennung.
w $0c00                ; Sektoren ab Track 1 / Sektor 1
b $00                 ; nach Bank 0 ab $0c00 laden.
b $02                 ; Zwei Sektoren nachladen.
b "DEMO",NULL         ; BOOT-Meldung...
b "",NULL              ; Kein Programm nachladen.

::start      jmp $0c00      ; Nachgeladene Sektoren ausführen.
```

Die Adresse »:start« ist relativ und folgt direkt den beiden, mit einem *NULL*-Byte abgeschlossenen Strings.

Hier ein weiterer Bootsektor als Speicher-Dump. Schreibt man diese Bytes direkt in ein SD2IEC-Diskimage ab Byte \$00:0000, dann wird daraus eine Boot-Diskette und es wird ein Programm "DEMO" geladen und gestartet.

```
$0b00  43 42 4d 00 00 00 00 44 45 4d 4f 00 00 a2 13 a0  CBM....DEMO....
$0b10  0b 4c a5 af 52 55 4e 22 44 45 4d 4f 22 00 00 00  ....RUN"DEMO"...
```

2.9.3 Systemadressen von GEOS 2.x

Die Einsprungadressen im GEOS-Kernal zu den Standard-Routinen sind unverändert und gelten ab sofort für alle Laufwerkstreiber (auch für Standard-1541-Laufwerke).

In allen Laufwerkstreibern findet sich am Beginn eine Sprungtabelle. Hier steht für jede Routine aber lediglich einen Zeiger auf die eigentliche Routine innerhalb des Laufwerkstreiber, der Einsprung in die Routine erfolgt daher über die GEOS-Sprungtabelle im Kernal mit *jsr Adresse* oder *jmp Adresse*:

Adresse	Intern	Routine	Beschreibung
\$c25c	\$9000	InitForIO	I/O-Bereich einschalten.
\$c250	\$9002	DoneWithIO	I/O-Bereich abschalten.
\$c232	\$9004	ExitTurbo	TurboDOS deaktivieren.
\$c235	\$9006	PurgeTurbo	TurboDOS entfernen.
\$c214	\$9008	EnterTurbo	TurboDOS aktivieren.
\$c2bc	\$900a	ChangeDiskDevice	Laufwerksadresse ändern.
\$c1e1	\$900c	NewDisk	Neue Diskette initialisieren.
\$c21a	\$900e	ReadBlock	Block von Diskette einlesen.
\$c220	\$9010	WriteBlock	Block auf Diskette schreiben.
\$c223	\$9012	VerWriteBlock	Block auf Diskette überprüfen.
\$c2a1	\$9014	OpenDisk	Diskette öffnen.
\$c1a4	\$9016	GetBlock	Block mit EnterTurbo/InitForIO einlesen.
\$c1e7	\$9018	PutBlock	Block mit EnterTurbo/InitForIO schreiben
\$c247	\$901a	GetDirHead	BAM von Diskette einlesen.
\$c24a	\$901c	PutDirHead	BAM auf Diskette schreiben.
\$c1f6	\$901e	GetFreeDirBlk	Freien Verzeichnisblock suchen.
\$c1db	\$9020	CalcBlksFree	Anzahl freie Blocks berechnen.
\$c2b9	\$9022	FreeBlock	Block in BAM freigeben.
\$c292	\$9024	SetNextFree	Nächsten freien Block in BAM belegen.
\$c2ad	\$9026	FindBAMBit	Block-Status in BAM abfragen.
\$c24d	\$9028	NxtBlkAlloc	Anzahl Blöcke in BAM reservieren.
\$c1fc	\$902a	BlkAlloc	Anzahl Blöcke ab Disk-Anfang in BAM reservieren.
\$c1de	\$902c	ChkDkGEOS	Diskette auf GEOS-Format testen.
\$c1ea	\$902e	SetGEOSDisk	Diskette in das GEOS-Format wandeln.

Eine Beschreibung der Routinen findet sich im **Teil B, Kapitel 12 ab Seite 282**.

Danach folgt eine Sprungtabelle, für die es im GEOS-Kernal keine Adresse gibt. D.h. diese Routinen werden direkt über ***jsr Adresse*** oder ***jmp Adresse*** innerhalb des aktuellen Laufwerkstreiber angesprochen:

Adresse	Routine	Beschreibung
\$9030	Get1stDirEntry	Ersten Verzeichniseintrag suchen.
\$9033	GetNxtDirEntry	Nächsten Verzeichniseintrags suchen.
\$9036	GetOPDPtr (GetBorderBlock)	Zeiger auf Borderblock einlesen.
\$9039	CreateNewDirBlk	Neuen Verzeichnisblock anlegen.
\$903c	GetDiskBlkBuf (GetBlock_dskBuf)	Block von Diskette nach diskBlkBuf einlesen.
\$903f	PutDiskBlkBuf (PutBlock_dskBuf)	Block in diskBlkBuf auf Diskette schreiben.
\$9042	TurboRoutine_r1	TurboDOS-Routine ausführen.
\$9045	GetDiskError	Fehlerstatus abfragen.
\$9048	AllocateBlock	Block in BAM reservieren..
\$904b	ReadLink	Linkbyte aus Block von Diskette einlesen.

Die beiden Routinen *TurboRoutine_r1* und *GetDiskError* werden ausschließlich von den internen TurboDOS-Routinen verwendet. Für Programmierer sind diese Routinen daher ohne Bedeutung, da der Umgang mit den TurboDOS-Routinen ausschließlich dem Laufwerkstreiber vorbehalten ist.

Eine Beschreibung der Routinen findet sich im **Teil B, Kapitel 12 ab Seite 282**.

2.9.4 Systemadressen von GEOS/MegaPatch

Im Anschluss an den Systembereich von GEOS 2.x folgt in allen Laufwerkstreibern in MP3 ein Kennbyte für den Laufwerkstyp. Der Wert entspricht dem Inhalt von *RealDrvType*, siehe dazu **Teil D, Kapitel 2.3 ab Seite 469**.

Adresse	Routine	Beschreibung
\$904e	DiskDrvType	Ersten Verzeichniseintrag suchen.
\$904f	DiskDrvVersion	Versionsangabe für Laufwerkstreiber (\$30)

Das Register *DiskDrvType* wurde vom GateWay-System übernommen: Hier findet man den Laufwerkstyp für das Laufwerk, wie er auch in *RealDrvType* abgelegt ist.

Im Register *DiskDrvVersion* findet man eine Versionsnummer des aktuellen Treibers. Man könnte meinen, das hier eine durchgehende Nummerierung aller Versionen erfolgt, dem ist leider nicht so. Jeder Programmierer setzt hier eigene Werte ein. Über dieses Register kann man also nicht feststellen ob ein Treiber bestimmte Funktionen unterstützt oder nicht.

Zuerst eine Übersicht der neuen Routinen. Teilweise sind diese schon in den Laufwerkstreibern des GateWay-Systems enthalten:

Adresse	Routine	Beschreibung
\$9050	OpenRootDir	NativeMode, Hauptverzeichnis öffnen.
\$9053	OpenSubDir	NativeMode, Unterverzeichnis öffnen.
\$9056	GetBAMBlock	NativeMode, BAM-Block von Diskette nach dir2Head einlesen.
\$9059	PutBAMBlock	NativeMode, BAM-Block in dir2Head auf Diskette schreiben.

Daneben gibt es noch ein paar neue Routinen:

Adresse	Routine	Beschreibung
\$905c	GetPDirEntry	CMD-Partitionsdaten mit EnterTurbo/InitForIO einlesen.
\$905f	ReadPDirEntry	CMD-Partitionsdaten einlesen.
\$9062	OpenPartition	Neue CMD-Partition mit EnterTurbo/InitForIO aktivieren.
\$9065	SwapPartition	Neue CMD-Partition aktivieren.
\$9068	GetPTypeData	Liste mit Partitionsformaten einlesen.
\$906b	SendFloppyCom	Befehl an Laufwerk senden.

Anschließend folgt eine Kennung für die erweiterten MP3-Laufwerkstreiber:

Adresse	Routine	Beschreibung
\$906e	DiskDrvTypeCode	GEOS/MegaPatch-Laufwerkstreiber.

Ab dieser Adresse findet sich die folgende Textkennung:

```
:DiskDrvTypeCode      b "MPDD3", NULL      ; Ab $906e
```

Die Einsprünge und Systemadressen sind in allen MP3-Laufwerkstreibern enthalten. Um nun festzustellen zu können ob der aktuelle Treiber die erweiterten Routinen unterstützt, wurde die Kennung "MPDD3" integriert.

Die Abkürzung steht für "MegaPatchDiskDriver Version 3".

Ab GEOS/MegaPatch V3.3r6 gibt es zusätzliche eingeführte Register und Einsprungsadressen. Um feststellen zu können ob ein solcher Treiber vorliegt, wurde eine weitere Kennung integriert:

Adresse	Routine	Beschreibung
\$9074	DiskDrvTypeExt	Erweiterter Laufwerkstreiber.

Ab dieser Adresse findet sich die folgende Textkennung:

```
:DiskDrvTypeExt      b "DDX", NULL      ; Ab $9074
```

Ist diese Kennung im Laufwerkstreiber vorhanden, dann unterstützt der Treiber zusätzliche Adressen und Funktionen, **Teil D, Kapitel 2.6 ab Seite 491**.

2.10 Symboltabellen und Makrodefinitionen

2.10.1 Symboltabelle "SymbTab.MP3"

Die folgende Symboltabelle ist eine Ergänzung zur Standard-Tabelle "SymbTab" des MegaAssembler und enthält neue Routinen und Symbole zu GEOS/MegaPatch:

```
; Symboltabelle für GEOS/MegaPatch3
; Revision 29.10.2022

; Erweiterte Systemadressen
:DskDrvBaseL      = $9f7e
:DskDrvBaseH      = $9f82
:doubleSideFlg    = $9f86
:drivePartData    = $9f8a
:RealDrvType      = $9f8e
:RealDrvMode      = $9f92
:RamBankInUse     = $9f96
:RamBankFirst     = $9fa6
:GEOS_RAM_TYP     = $9fa8
:MP3_64K_SYSTEM   = $9fa9
:MP3_64K_DATA     = $9faa
:MP3_64K_DISK     = $9fab
:Flag_Optimize    = $9fac
:millenium        = $9fad
:Flag_LoadPrnt    = $9fae
:PrntFileNameRAM  = $9faf
:Flag_Spooler     = $9fc0
:Flag_SpoolMinB   = $9fc1
:Flag_SpoolMaxB   = $9fc2
:Flag_SpoolADDR   = $9fc3
:Flag_SpoolCount  = $9fc6
:Flag_SplCurDok   = $9fc7
:Flag_SplMaxDok   = $9fc8
:Flag_TaskAktiv   = $9fc9
:Flag_TaskBank    = $9fca
:Flag_ExtRAMinUse = $9fcb
:Flag_ScrSvCnt    = $9fcc
:Flag_ScrSaver    = $9fcd
:Flag_CrsrRepeat  = $9fce
:BackScrPattern   = $9fcf
:Flag_SetColor    = $9fd0
:Flag_ColorDBox   = $9fd1
:Flag_IconMinX    = $9fd2
:Flag_IconMinY    = $9fd3
:Flag_IconDown    = $9fd4
; :Flag_DBoxType    = $9fd5      ;Used by kernal only.
; :Flag_GetFiles    = $9fd6      ;Used by kernal only.
:DB_GFileType     = $9fd7
:DB_GFileClass    = $9fd8
:DB_GetFileEntry  = $9fda
:DB_StdBoxSize    = $9fdb
:Flag_SetMLine    = $9fe1
:Flag_MenuStatus  = $9fe2
```

```

:DM_LastEntry          = $9fe3
:DM_LastNumEntry       = $9fe9
:MP3_COLOR_DATA        = $9fea

; Systemadressen der Farbtabelle
:C_Balken              = $9fea
:C_Register            = $9feb
:C_RegisterOff         = $9fec
:C_RegisterBack        = $9fed
:C_Mouse              = $9fee
:C_DBoxTitel           = $9fef
:C_DBoxBack            = $9ff0
:C_DBoxDIcon           = $9ff1
:C_FBoxTitel           = $9ff2
:C_FBoxBack            = $9ff3
:C_FBoxDIcon           = $9ff4
:C_FBoxFiles           = $9ff5
:C_WinTitel            = $9ff6
:C_WinBack             = $9ff7
:C_WinShadow           = $9ff8
:C_WinIcon             = $9ff9
:C_PullDMenu           = $9ffa
:C_InputField          = $9ffb
:C_InputFieldOff       = $9ffc
:C_GEOS_BACK           = $9ffd
:C_GEOS_FRAME          = $9ffe
:C_GEOS_MOUSE          = $9fff

; Speichererweiterungen
:RAM_SCPU              = $10
:RAM_BBG               = $20
:RAM_GEORAM            = $20
:RAM_REU               = $40
:RAM_RL                = $80

; Flag_SetColor: Dialogbox-Farben setzen
:SET_DBOXCOL_OFF       = $00
:SET_DBOXCOL_STD       = $40
:SET_DBOXCOL_ON        = $80

; Zusätzliche Symbole für DoDlgBox
:DBOXCOLON             = %01000000
:DBOXCOLOFF            = %00000000
:DRIVE                 = $07
:DUMMY                 = $08
:DBUSRFILES            = $09
:DBSETCOL              = $0a
:DBSELECTPART          = %10000000
:DBSETDRVICON          = %01000000
:DBOXDRVA              = $88
:DBOXDRVB              = $89
:DBOXDRVC              = $8a
:DBOXDRVD              = $8c

```

```
; Neue Kernalroutinen
:i_UserColor           = $c0dc
:i_ColorBox           = $c0df
:DirectColor          = $c0e2
:RecColorBox          = $c0e5
:GetBackScreen        = $c0e8
:ResetScreen          = $c0eb
:GEOS_InitSystem      = $c0ee
:PutKeyInBuffer       = $c0f1
:SCPU_Pause           = $c0f4

; SuperCPU-Optimierung für GEOS
; :SCPU_OptOn          = $c0f7
; :SCPU_OptOff        = $c0fa
; :SCPU_SetOpt         = $c0fd

; Register-Menu
:BOX_USER             = $01
:BOX_USER_VIEW        = $02
:BOX_USEROPT          = $03
:BOX_USEROPT_VIEW     = $04
:BOX_FRAME            = $05
:BOX_ICON             = $06
:BOX_ICON_VIEW        = $07
:BOX_OPTION           = $08
:BOX_OPTION_VIEW      = $09
:BOX_STRING           = $0a
:BOX_STRING_VIEW      = $0b
:BOX_NUMERIC          = $0c
:BOX_NUMERIC_VIEW     = $0d
:NUMERIC_LEFT         = %00000000
:NUMERIC_RIGHT        = %10000000
:NUMERIC_SETSPC       = %00000000
:NUMERIC_SET0         = %01000000
:NUMERIC_BYTE         = %00000000
:NUMERIC_WORD         = %00100000
:USE_COLOR_INPUT      = $ff
:USE_COLOR_REG        = $ee
:NO_OPT_UPDATE        = $00

; Registermenü-Routinen
:DoRegister           = $6d00
:ExitRegisterMenu     = $6d03
:RegisterInitMenu     = $6d06
:RegisterUpdate       = $6d09
:RegisterAllOpt       = $6d0c
:RegisterNextOpt      = $6d0f
:RegDrawOptFrame     = $6d12
:RegClrOptFrame      = $6d15
:RegisterSetFont      = $6d18
:RegisterAktiv        = $6d1b
```

```

; Systemadressen Registermenü
:SetADDR_Register      = $cfe6          ;Zeiger auf Registermenü
:LD_ADDR_REGISTER      = $6d00          ;Ladeadresse Registermenü

; Erweiterte Adressen im Laufwerkstreiber
:OpenRootDir           = $9050
:OpenSubDir            = $9053
:GetBAMBlock           = $9056
:PutBAMBlock           = $9059
:GetPDirEntry          = $905c
:ReadPDirEntry         = $905f
:OpenPartition         = $9062
:SwapPartition         = $9065
:GetPTypeData          = $9068
:SendFloppyCom         = $906b
:DiskDrvTypeExt        = $9074
:DDRV_EXT_DATA1        = $907a
:DDRV_EXT_DATA2        = $907b
:InitForDskDvOp        = $907c
:DoneWithDskDvOp       = $907f
:dir3Head              = $9c80

; Verschiedene Symbole
:OS_VARS               = $8000          ;OS variable base
:MP3_CODE              = $c014
:DDRV_CODE             = $906e
:DDX_CODE              = $9074
:BASE_AUTO_BOOT        = $5000          ;Ladeadresse der AutoBoot-Routine.
:SIZE_AUTO_BOOT        = $0500          ;max. Größe der AutoBoot-Routine.

; Erweiterte Diskettenfehlermeldungen
:NO_ERROR              = $00
; :NO_BLOCKS           = $01
:INV_TRACK             = $02
; :INSUFF_SPACE        = $03
:FULL_DIRECTORY        = $04
:FILE_NOT_FOUND        = $05
:BAD_BAM               = $06
; :UNOPENED_VLIR      = $07
; :INV_RECORD          = $08
; :OUT_OF_RECORDS      = $09
; :STRUCT_MISMAT       = $0a          ;In TopSym definiert.
:BFR_OVERFLOW          = $0b
:CANCEL_ERR            = $0c
:DEV_NOT_FOUND         = $0d
; :INCOMPATIBLE        = $0e          ;In TopSym definiert.
; :HDR_NOT_THERE       = $20
:NO_SYNC               = $21
; :DBLK_NOT_THERE      = $22
; :DAT_CHKSUM_ERR      = $23
:WR_VER_ERR            = $25
:WR_PR_ON              = $26
; :HDR_CHKSUM_ERR      = $27
; :DSK_ID_MISMAT       = $29
; :BYTE_DEC_ERR        = $2e

```

```
; :NO_PARTITION           = $30
; :PART_FORMAT_ERR       = $31
; :ILLEGAL_PARTITION     = $32
; :NO_PART_FD_ERR        = $33
; :ILLEGAL_DEVICE        = $40
:NO_FREE_RAM             = $60
:DOS_MISMATCH            = $73

; Definition der Laufwerkstypen
:ST_DMODES               = %00000111
; :DRIVE_MODES           = %00000111
:Drv1541                 = $01
:Drv1571                 = $02
:Drv1581                 = $03
:DrvIECBNM              = $04
:DrvSD2IEC               = $04
:DrvNative               = $04
:DrvPCDOS               = $05
:Drv81DOS               = $05

:DrvShadow               = %01000000
:DrvShadow1541           = DrvShadow ! Drv1541
; :DrvShadow1571         = DrvShadow ! Drv1571
; :DrvShadow1581         = DrvShadow ! Drv1581
; :DrvShadowNM           = DrvShadow ! DrvNative

:DrvRAM                  = %10000000
:DrvRAM1541              = DrvRAM ! Drv1541
:DrvRAM1571              = DrvRAM ! Drv1571
:DrvRAM1581              = DrvRAM ! Drv1581
:DrvRAMNM                = DrvRAM ! DrvNative

:DrvCREU                 = %10100000
:DrvRAMNM_CREU           = DrvCREU ! DrvNative
:DrvGRAM                 = %10110000
:DrvRAMNM_GRAM           = DrvGRAM ! DrvNative
:DrvSCPU                 = %11000000
:DrvRAMNM_SCPU           = DrvSCPU ! DrvNative

:DrvFD                   = %00010000
:DrvFD41                 = DrvFD ! Drv1541
:DrvFD71                 = DrvFD ! Drv1571
:DrvFD81                 = DrvFD ! Drv1581
:DrvFD2                  = DrvFD
:DrvFD4                  = DrvFD
:DrvFDNM                 = DrvFD ! DrvNative
:DrvFDDOS                = DrvFD ! DrvPCDOS

:DrvHD                   = %00100000
:DrvHD41                 = DrvHD ! Drv1541
:DrvHD71                 = DrvHD ! Drv1571
:DrvHD81                 = DrvHD ! Drv1581
:DrvHDNM                 = DrvHD ! DrvNative
```

```

:DrvRAMLink          = %00110000
:DrvRL41             = DrvRAMLink ! Drv1541
:DrvRL71             = DrvRAMLink ! Drv1571
:DrvRL81             = DrvRAMLink ! Drv1581
:DrvRLNM             = DrvRAMLink ! DrvNative

:DrvCMD              = %00110000

; Definition der Laufwerks-Modi
:SET_MODE_PARTITION  = %10000000
:SET_MODE_SUBDIR     = %01000000
:SET_MODE_FASTDISK   = %00100000
:SET_MODE_SRAM       = %00010000
:SET_MODE_CRAM       = %00001000
:SET_MODE_GRAM       = %00000100
:SET_MODE_SD2IEC     = %00000010

; CBM-Dateitypen
:FMODE_CLOSED        = %10000000
:FMODE_WRPROT        = %01000000
:FTYPE_MODES         = %00000111
:FTYPE_DEL           = $00
:FTYPE_SEQ           = $01
:FTYPE_PRG           = $02
:FTYPE_USR           = $03
:FTYPE_REL           = $04
:FTYPE_DIR           = $06

; Sonstige C64-Systemadressen
:zpage               = $0000

```

2.10.2 Symboltabelle "SymbTab.IO"

```

; Adressen im I/O-Bereich
; Revision 29.10.2022

:rasreq              = $d012      ;Current raster scanline.
:extclr              = $d020      ;Border color.
:bakclr0             = $d021      ;Background color.
:mob0clr             = $d027      ;Farbe Sprite #0.
:mob1clr             = $d028      ;Farbe Sprite #1.
:mob2clr             = $d029      ;Farbe Sprite #2.
:CIA_PRA              = $dc00      ;CIA Reg. DataPort A, Bit PA0 to PA7.
:CIA_PRB              = $dc01      ;CIA Reg. DataPort B, Bit PB0 to PB7.
:CIA_TOD10            = $dc08      ;CIA TOD 1/10 Sekunden.
:CIA_TODSEC           = $dc09      ;CIA TOD Sekunden.
:CIA_TODMIN           = $dc0a      ;CIA TOD Minuten.
:CIA_TODHR            = $dc0b      ;CIA TOD Stunde.

```

2.10.3 Symboltabelle "SymbTab.ROM"

```
; Einsprünge im C64-Kernal
; Version 29.10.2022

:IOINIT          = $fda3      ;Reset: CIA.
:CINT            = $ff81      ;Reset: Timer, IO, PAL/NTSC, Bildschirm.
; :IOINIT        = $ff84      ;Reset: CIA.
:SETMSG          = $ff90      ;Dateiparameter definieren.
:SECOND          = $ff93      ;Sekundär-Adresse nach LISTEN senden.
:TKSA            = $ff96      ;Sekundär-Adresse nach TALK senden.
:ACPTR           = $ffa5      ;Byte-Eingabe vom IEC-Bus.
:CIOUT           = $ffa8      ;Byte-Ausgabe auf IEC-Bus.
:UNTALK          = $ffab      ;UNTALK-Signal auf IEC-Bus senden.
:UNLSN           = $ffae      ;UNLISTEN-Signal auf IEC-Bus senden.
:LISTEN          = $ffb1      ;LISTEN-Signal auf IEC-Bus senden.
:TALK            = $ffb4      ;TALK-Signal auf IEC-Bus senden.
:SETLFS          = $ffba      ;Dateiparameter setzen.
:SETNAM          = $ffbd      ;Dateiname setzen.
:OPENCHN         = $ffc0      ;Datei öffnen.
:CLOSE           = $ffc3      ;Datei schließen.
:CHKIN           = $ffc6      ;Eingabefile setzen.
:CKOUT           = $ffc9      ;Ausgabefile setzen.
:CLRCHN          = $ffcc      ;Standard-I/O setzen.
:BSOUT           = $ffd2      ;Zeichen ausgeben.
:LOAD            = $ffd5      ;Datei laden.
:GETIN           = $ffe4      ;Tastatur-Eingabe.
:CLALL           = $ffe7      ;Alle Kanäle schließen.

;*** Einsprünge im RAMLink-Kernal.
:EN_SET_REC      = $e0a9      ;Enable RAMLink, set REC page.
:RL_HW_EN        = $e0b1      ;Enable RAMLink, turn off IRQ.
:SET_REC_IMG     = $fe03      ;Set REC page.
:EXEC_REC_REU    = $fe06      ;Exec according to REU register.
:EXEC_REC_SEC    = $fe09      ;Exec according to sector register.
:RL_HW_DIS       = $fe0c      ;Disable RAMLink, turn IRQ on.
:RL_HW_DIS2      = $fe0f      ;Disable RAMLink, leave IRQ off.
:EXEC_REU_DIS    = $fe1e      ;Exec REU, Disable RL, IRQ on.
:EXEC_SEC_DIS    = $fe21      ;Exec sector, Disable RL, IRQ on.
```


2.10.4 Erweiterte Symboltabelle "ExtSym.MP3"

```
; Erweiterte Symboltabelle für GEOS/MegaPatch
; Revision 29.10.2022

; Symbole für Inhalt der GEOS-Speicherbank
; (Erste Speicherbank im GEOS-DACC)
; R1_SIZE_MOVEDATA      = $7900      ;MoveData-Transfer-Bereich
; :R1_SIZE_SYS_VAR1     = $0500      ;Kernal-Variablen
; :R1_SIZE_REBOOT       = $0500      ;ReBoot-Routine
; R1_SIZE_DSKDEV_A      = $0d80      ;Laufwerkstreiber A:
; R1_SIZE_DSKDEV_B      = $0d80      ;Laufwerkstreiber B:
; R1_SIZE_DSKDEV_C      = $0d80      ;Laufwerkstreiber C:
; R1_SIZE_DSKDEV_D      = $0d80      ;Laufwerkstreiber D:
; :R1_SIZE_SYS_PRG1     = $0280      ;Kernal $9D80-$9FFF
; :R1_SIZE_SYS_PRG2     = $10c0      ;Kernal $BF40-$CFFF
; :R1_SIZE_SYS_PRG3     = $3000      ;Kernal $D000-$DFFF
; :R1_SIZE_RB00TMSE     = $003f      ;Aktuelles Mauszeiger-Icon
; :R1_SIZE_SYS_BBG1     = $0100      ;DoRAMOp-Zusatz für BBGRAM
; :R1_SIZE_SYS_BBG2     = $0100      ;DoRAMOp-Zusatz für BBGRAM

; R1_ADDR_MOVEDATA      = $0000
; :R1_ADDR_SYS_VAR1     = $7900
; :R1_ADDR_REBOOT       = $7e00
; R1_ADDR_DSKDEV_A      = $8300
; R1_ADDR_DSKDEV_B      = $9080
; R1_ADDR_DSKDEV_C      = $9e00
; R1_ADDR_DSKDEV_D      = $ab80
; :R1_ADDR_SYS_PRG1     = $b900
; :R1_ADDR_SYS_PRG2     = $bb80
; :R1_ADDR_SYS_PRG3     = $cc40
; :R1_ADDR_RB00TMSE     = $fc40
; :R1_ADDR_SYS_BBG1     = $fe00
; :R1_ADDR_SYS_BBG2     = $ff00
```

```

; Symbole für die Speicherbank mit den
; erweiterten Routinen in MegaPatch
; (Letzte Speicherbank im GEOS-DACC)
:R2_SIZE_REGISTER      = $0c00      ;Registermenü-Routine
:R2_SIZE_ENTER_DT      = $0200      ;EnterDesktop-Routine
:R2_SIZE_PANIC         = $0100      ;Neue PANIC!-Box
:R2_SIZE_TOBASIC       = $0200      ;Neue ToBasic-Routine
:R2_SIZE_GETNXDAY      = $0080      ;Nächsten Tag berechnen
:R2_SIZE_DOALARM       = $0080      ;Weckzeit anzeigen
:R2_SIZE_GETFILES      = $1c00      ;Neue Dateiauswahlbox
:R2_SIZE_GFILDATA      = $0180      ;GetFiles-Subroutine
:R2_SIZE_GFILMENU      = $0380      ;GetFiles-Subroutine
:R2_SIZE_DB_SCREEN     = $0300      ;Dialogboxbildschirm laden/speichern
:R2_SIZE_DB_COLOR      = 25*40      ;Dialogboxbildschirm: Farbe
:R2_SIZE_DB_GRAFX      = 25*40*8    ;Dialogboxbildschirm: Grafik
:R2_SIZE_GETBSCRN      = $0100      ;Hintergrundbild einlesen
:R2_SIZE_BS_COLOR      = 25*40      ;Hintergrundbild: Farbe
:R2_SIZE_BS_GRAFX      = 25*40*8    ;Hintergrundbild: Grafik
:R2_SIZE_SCRSAVER      = $1c00      ;Bildschirmschoner-Routine
:R2_SIZE_SS_COLOR      = 25*40      ;Bildschirmschoner: Farbe
:R2_SIZE_SS_GRAFX      = 25*40*8    ;Bildschirmschoner: Grafik
:R2_SIZE_SPOOLER       = $1508      ;Spooler-Menü
:R2_SIZE_PRNSPHDR      = $0100      ;Header für Druckerspooler-Treiber
:R2_SIZE_PRNSPOOL      = $0640      ;Druckerspooler-Treiber
:R2_SIZE_PRNTHDR       = $0100      ;Header für Drucker-Treiber
:R2_SIZE_PRINTER       = $0640      ;Drucker-Treiber
:R2_SIZE_TASKMAN       = $2000      ;Größe des TaskSwitchers

:R2_ADDR_REGISTER      = $0000
:R2_ADDR_ENTER_DT      = (R2_ADDR_REGISTER + R2_SIZE_REGISTER)
:R2_ADDR_PANIC         = (R2_ADDR_ENTER_DT + R2_SIZE_ENTER_DT)
:R2_ADDR_TOBASIC       = (R2_ADDR_PANIC + R2_SIZE_PANIC)
:R2_ADDR_GETNXDAY      = (R2_ADDR_TOBASIC + R2_SIZE_TOBASIC)
:R2_ADDR_DOALARM       = (R2_ADDR_GETNXDAY + R2_SIZE_GETNXDAY)
:R2_ADDR_GETFILES      = (R2_ADDR_DOALARM + R2_SIZE_DOALARM)
:R2_ADDR_GFILDATA      = (R2_ADDR_GETFILES + R2_SIZE_GETFILES)
:R2_ADDR_GFILMENU      = (R2_ADDR_GFILDATA + R2_SIZE_GFILDATA)
:R2_ADDR_DB_SCREEN     = (R2_ADDR_GFILMENU + R2_SIZE_GFILMENU)
:R2_ADDR_DB_COLOR      = (R2_ADDR_DB_SCREEN+ R2_SIZE_DB_SCREEN)
:R2_ADDR_DB_GRAFX      = (R2_ADDR_DB_COLOR + R2_SIZE_DB_COLOR)
:R2_ADDR_GETBSCRN      = (R2_ADDR_DB_GRAFX + R2_SIZE_DB_GRAFX)
:R2_ADDR_BS_COLOR      = (R2_ADDR_GETBSCRN + R2_SIZE_GETBSCRN)
:R2_ADDR_BS_GRAFX      = (R2_ADDR_BS_COLOR + R2_SIZE_BS_COLOR)
:R2_ADDR_SCRSAVER      = (R2_ADDR_BS_GRAFX + R2_SIZE_BS_GRAFX)
:R2_ADDR_SS_COLOR      = (R2_ADDR_SCRSAVER + R2_SIZE_SCRSAVER)
:R2_ADDR_SS_GRAFX      = (R2_ADDR_SS_COLOR + R2_SIZE_SS_COLOR)
:R2_ADDR_SPOOLER       = (R2_ADDR_SS_GRAFX + R2_SIZE_SS_GRAFX)
:R2_ADDR_PRNSPHDR      = (R2_ADDR_SPOOLER + R2_SIZE_SPOOLER)
:R2_ADDR_PRNSPOOL      = (R2_ADDR_PRNSPHDR + R2_SIZE_PRNSPHDR)
:R2_ADDR_PRNTHDR       = (R2_ADDR_PRNSPOOL + R2_SIZE_PRNSPOOL)
:R2_ADDR_PRINTER       = (R2_ADDR_PRNTHDR + R2_SIZE_PRNTHDR)
:R2_ADDR_TASKMAN_B     = (R2_ADDR_PRINTER + R2_SIZE_PRINTER)

; :R2_ADDR_TASKMAN      = $4000      ;Adr. TaskManager
; :R2_ADDR_TASKMAN_E    = $6000      ;Adr. TaskManager während GEOS.Editor

```

```

; Symbole für die Speicherbank mit den
; Zwischenspeichern für GEOS/MegaPatch
; (Vorletzte Speicherbank im GEOS-DACC)
; :R3_SIZE_SWAPFILE      = $7c00          ;Größe der Auslagerungsdatei
; :R3_SIZE_FNAMES        = $1200          ;Puffer für Dateinamen
; :R3_SIZE_AUTOBBUF      = SIZE_AUTO_BOOT ;Puffer AutoBoot-Routine
; :R3_SIZE_REGMEMBUF     = R2_SIZE_REGISTER;Puffer Registermenü
; :R3_SIZE_ZEROPBUF      = $0400          ;Puffer Zeropage
; :R3_SIZE_OSVARBUF      = $0c00          ;Puffer OS_VARS
; :R3_SIZE_MPVARBUF      = $0050          ;Puffer OS_VAR_MP
; :R3_SIZE_SP_COLOR      = 25*40          ;Puffer Druckerspooler/Farbe.
; :R3_SIZE_SP_GRAFX      = 25*40*8        ;Puffer Druckerspooler/Grafik
; :R3_SIZE_SPOOLDAT      = 640 + 80 + 1920 ;Puffer Druckerspooler/Daten
; :R3_SIZE_PRNSPLTMP     = $0640          ;Temp. Kopie Spooler-Treiber

; :R3_ADDR_SWAPFILE      = $0000
; :R3_ADDR_FNAMES        = (R3_ADDR_SWAPFILE + R3_SIZE_SWAPFILE)
; :R3_ADDR_AUTOBBUF      = (R3_ADDR_FNAMES + R3_SIZE_FNAMES)
; :R3_ADDR_REGMEMBUF     = (R3_ADDR_AUTOBBUF + R3_SIZE_AUTOBBUF)
; :R3_ADDR_ZEROPBUF      = (R3_ADDR_REGMEMBUF + R3_SIZE_REGMEMBUF)
; :R3_ADDR_OSVARBUF      = (R3_ADDR_ZEROPBUF + R3_SIZE_ZEROPBUF)
; :R3_ADDR_MPVARBUF      = (R3_ADDR_OSVARBUF + R3_SIZE_OSVARBUF)
; :R3_ADDR_SP_COLOR      = (R3_ADDR_MPVARBUF + R3_SIZE_MPVARBUF)
; :R3_ADDR_SP_GRAFX      = (R3_ADDR_SP_COLOR + R3_SIZE_SP_COLOR)
; :R3_ADDR_SPOOLDAT      = (R3_ADDR_SP_GRAFX + R3_SIZE_SP_GRAFX)
; :R3_ADDR_PRNSPLTMP     = (R3_ADDR_SPOOLDAT + R3_SIZE_SPOOLDAT)
; :R3_ADDR_END_MP3       = (R3_ADDR_PRNSPLTMP + R3_SIZE_PRNSPLTMP)

```

```
; Symbole für die Ladeadressen der
; erweiterten Routinen in GEOS/MegaPatch
:LD_ADDR_NEWBSCRN      = $7800
; :LD_ADDR_REGISTER    = PRINTBASE - R2_SIZE_REGISTER
:LD_ADDR_ENTER_DT      = diskBlkBuf - R2_SIZE_ENTER_DT
; :LD_ADDR_PANIC        = diskBlkBuf
; :LD_ADDR_TOBASIC      = DISK_BASE - R2_SIZE_TOBASIC
; :LD_ADDR_GETNXDAY      = diskBlkBuf
; :LD_ADDR_DOALARM       = diskBlkBuf
; :LD_ADDR_GETFILES      = BACK_SCR_BASE
; :LD_ADDR_GFILDATA      = dlgBoxRamBuf + 0
; :LD_ADDR_GFILPART      = dlgBoxRamBuf + 9
; :LD_ADDR_GFILMENU      = diskBlkBuf
; :LD_ADDR_GFILICON      = LD_ADDR_GFILMENU + 3
; :LD_ADDR_GFILFBOX      = LD_ADDR_GFILMENU + 6
; :LD_ADDR_DBOXICON      = LD_ADDR_GFILMENU + 9
; :DB_FNAME_BUF          = LD_ADDR_GETFILES - R3_SIZE_FNAMES
; :DB_PDATA_BUF          = LD_ADDR_GETFILES - 256
; :LD_ADDR_DB_SCREEN     = diskBlkBuf
; :DB_SCREEN_SAVE       = LD_ADDR_DB_SCREEN + 0
; :DB_SCREEN_LOAD       = LD_ADDR_DB_SCREEN + 3
; :LD_ADDR_TASKMAN       = $4000
; :LD_ADDR_INIT_GEOS     = diskBlkBuf
:LD_ADDR_SCRSAVER      = OS_VARS - R2_SIZE_SCRSAVER
:LD_ADDR_SCRSVINIT     = LD_ADDR_SCRSAVER + 3
:LD_ADDR_GETBSCRN      = diskBlkBuf
; :LD_ADDR_SPOOLER      = $4000

; Symbole für den Zugriff auf die
; erweiterten Routinen in GEOS/MegaPatch
:SetADDR_TaskMan        = $cfed
; :SetADDR_Register     = $cfe6           ; In TopSym.MP3 definiert
:SetADDR_EnterDT        = $cfe3
; :SetADDR_ToBASIC      = $cfe0
; :SetADDR_PANIC        = $cfdd
; :SetADDR_GetNxDay      = $cfda
; :SetADDR_DoAlarm       = $cfd7
; :SetADDR_GetFiles      = $cfd4
; :SetADDR_GFilData      = $cfd1
; :SetADDR_GFilMenu      = $cfce
; :SetADDR_DB_SCRN       = $cfcb
; :SetADDR_DB_GRFx       = $cfc8
; :SetADDR_DB_COLS       = $cfc5
:SetADDR_BackScrN       = $cfc2
:SetADDR_ScrSaver       = $cfbf
; :SetADDR_Spooler      = $cfbc
; :SetADDR_PrnSpool      = $cfb9
; :SetADDR_PrnSpHdr      = $cfb6
; :SetADDR_Printer       = $cfb3
; :SetADDR_PrntHdr       = $cfb0
```

2.10.5 Makrodefinitionen "TopMac.MP3"

```
; Makrodefinitionen für GEOS/MegaPatch3
; Revision 29.10.2022
```

```
; *****
;
; *                               *
; *   ClrB   Adresse               *
; *                               *
; *   Löscht Byte in Adresse       *
; *                               *
; *****
```

```
:ClrB      m
           lda    #$00
           sta    $0
           /
```

```
; *****
;
; *                               *
; *   ClrW   Adresse               *
; *                               *
; *   Löscht Word ab Adresse       *
; *                               *
; *****
```

```
:ClrW      m
           lda    #$00
           sta    $0
           sta    $0 +1
           /
```

```
; *****
;
; *                               *
; *   AddVBW Wert, Adresse         *
; *                               *
; *   Addiert Byte-Wert zu Adresse *
; *                               *
; *****
```

```
:AddVBW    m
           lda    #$0
           clc
           adc    $1
           sta    $1
           bcc    :Exit
           inc    $1+1
```

```
::Exit
```

```
/
```

Anhang K

Auf den folgenden Seiten findet sich eine Kurzreferenz der GEOS-Routinen mit den für die Verwendung erforderlichen Parametern. Die genaue Funktionsweise kann im **Teil B ab Kapitel 2 / Seite 173** nachgelesen werden.

Die Kurzreferenz orientiert sich am Aufbau von Teil B und am "Official GEOS Programmers Reference Guide" von Berkeley Softworks.

Die veränderten Register wurden zusätzlich mit "The Hitchhikers Guide to GEOS" von 1988 abgeglichen und müssen nicht den tatsächlich veränderten Registern der einzelnen Routinen entsprechen: Die Angabe entspricht eher einer Liste von Registern, welche von der entsprechenden Routine verändert werden könnten.

Die Angaben in den beiden Referenzen weichen aber teilweise voneinander ab, auf Unterschiede wird dann in der Beschreibung hingewiesen.

GEOS-Routinen, die nur unter GEOS/MegaPatch 64/128 verfügbar sind, werden mit "MP3" in der Überschrift gekennzeichnet.

Die Nummerierung innerhalb der Kurzreferenz orientiert sich am bestehenden Aufbau von **Teil B, Kapitel 2 ab Seite 204** und **Teil D, Kapitel 2.6 ab Seite 485**.

Übersicht Kurzreferenz

K.1	Systemübersicht	539
K.2	Menüroutinen	556
K.3	Dialogboxroutinen	560
K.4	Grafikroutinen	563
K.5	Textroutinen	578
K.6	Mausroutinen	586
K.7	Spriteroutinen	590
K.8	Speicherverwaltungsroutinen	592
K.9	Rechenroutinen	598
K.10	Prozessroutinen	601
K.11	I/O-Routinen	603
K.12	Diskettenroutinen	605
K.13	Druckerrountinen	639
K.14	Die restlichen Routinen	641
K.15	Das Registermenü von GEOS/MegaPatch	646

K.1 Systemübersicht

K.1.1 Memory-Map

Speicherbelegung von GEOS64, für GEOS128 in Bank 1=FrontRAM:

Bereich	Funktion
\$0000	6510 Datenrichtungsregister
\$0001	6510 I/O-Register
\$0002 - \$0021	Zeropage für GEOS und Applications, Register r0 bis r15
\$0022 - \$004f	Zeropage für GEOS
\$0058 - \$006f	Zeropage für Applications, Register u0 bis u11 (CC65)
\$0070 - \$007f	Zeropage für Applications, Register a2 bis a9
\$0080 - \$00fa	Zeropage für C64-Kernal und BASIC-Routinen (sharedZPage)
\$00fb - \$00fe	Zeropage für Applications, Register a0 und a1
\$00ff	Zeropage, nicht verwendet
\$0100 - \$01ff	Prozessorstack
\$0200 - \$03ff	Speicher für C64-Kernaldaten. Teilweise durch DeskTop64 V2 belegt (APP_LVAR und APP_LRAM)
\$0200 - \$0258	Variablenspeicher für Anwendungen (APP_LVAR) Hinweis: Darf durch DeskAccessories nicht verändert werden!
\$0334 - \$03ff	Erweiterter Anwendungsspeicher (APP_LRAM)
\$0400 - \$7fff	Anwendungsspeicher (APP_RAM)
\$5000 - \$5fff	GEOS-Autoboot-Routine oder Anwendungsspeicher
\$6000 - \$7f3f	Hintergrundgrafik (BACK_SCR_BASE)
\$7900 - \$7f3f	Druckertreiber (PRINTBASE)
\$7f40 - \$7fff	Anwendungsspeicher (APP_VAR)
\$8000 - \$89ff	Speicher für Zugriff auf Disketten, GEOS-Variablen (OS_BASE)
\$8a00 - \$8bff	Sprite-Grafikdaten (SPRITE_PICS)
\$8c00 - \$8fe7	Farbdaten für Bildschirmgrafik (COLOR_MATRIX)
\$8fe8 - \$8ff7	Speicher für GEOS-Kernal (sysApplData)
\$8ff8 - \$8fff	Sprite-Zeiger
\$9000 - \$9d7f	Laufwerkstreiber (DISK_BASE)
\$9d80 - \$9fff	Speicher für GEOS-Kernal (OS_LOW)
\$a000 - \$bfff	ROM: C64-BASIC-Routinen
\$a000 - \$bf3f	Vordergrundgrafik (SCREEN_BASE)
\$bf40 - \$bf7e	Grafikdaten für Mauszeiger
\$bf7f - \$bfff	Speicher für GEOS-Kernal
\$c000 - \$cfff	Speicher für GEOS-Kernal
\$d000 - \$dfff	Speicher für GEOS-Kernal und I/O-Bereich
\$e000 - \$ffff	ROM: C64-Kernalroutinen
\$e000 - \$fe7f	Speicher für GEOS-Kernal
\$fe80 - \$fff9	Eingabetreiber (MOUSE_BASE)
\$fffa - \$ffff	6510 NMI-, IRQ- und Reset-Vektor

Unter GEOS128 gibt es einige Bereiche die abweichend belegt sind:

Bereich	Funktion
\$d8c0 - \$d9bf	Infoblock für den aktuellen Druckertreiber
\$d9c0 - \$dfff	Aktueller Druckertreiber
\$fd00 - \$fe7f	Hier liegt unter GEOS128 der aktuelle Eingabetreiber. Die Sprungtabelle ab MOUSE_BASE (\$fe80) verweist auf den Bereich ab MOUSE_JMP (\$fd00).

Für GEOS128 folgt hier die Speicherbelegung im BackRAM:

Bereich	Funktion
\$0000 - \$03ff	1Kb Common Area
\$0400 - \$1fff	Speicher für GEOS-Kernal, u.a. Daten für SoftSprite-Handler
\$2000 - \$7fff	Zwischenspeicher für Swapfile beim Start von DeskAccessories
\$8000 - \$abff	Nicht verwendet, vermutlich frei für Applications
\$8000 - \$84ff	Nur MP3: Wird von GEOS128.Editor für SD-Tools verwendet
\$ac00 - \$bfff	Verzeichnis-Cache für 1541/1571-Laufwerkstreiber
\$c000 - \$cfff	Weitere Routinen des GEOS128-Kernal
\$d000 - \$dd7f	Laufwerkstreiber für Laufwerk B wenn keine REU vorhanden ist
\$e000 - \$ffff	Weitere Routinen des GEOS128-Kernal

Wenn eine REU vorhanden ist, dann ist Bank#0 wie folgt belegt:

Bereich	Funktion
\$0000 - \$78ff	C64: Zwischenspeicher für MoveData mit DMA
\$0000 - \$38ff	C128: Zwischenspeicher für MoveData mit DMA
\$3900 - \$78ff	C128: GEOS-Kernal von \$e000 bis \$ffff aus BackRAM für RBoot
\$7900 - \$7dff	GEOS-Variablen von \$8400 bis \$88ff für RBoot
\$7e00 - \$82ff	RBoot-Routine
\$8300 - \$907f	Laufwerkstreiber für Laufwerk A
\$9080 - \$9dff	Laufwerkstreiber für Laufwerk B
\$9e00 - \$ab7f	Laufwerkstreiber für Laufwerk C
\$ab80 - \$b8ff	Laufwerkstreiber für Laufwerk D
\$b900 - \$bb7f	GEOS-Kernal von \$9d80 bis \$9fff für RBoot
\$bb80 - \$cc3f	GEOS-Kernal von \$bf40 bis \$cfff für RBoot
\$cc40 - \$fc3f	GEOS-Kernal von \$d000 bis \$ffff für RBoot
\$fc40 - \$fc7f	Mauszeiger von \$84c1 bis \$84ff (mousePicData) für RBoot

K.1.2 GEOS-Systemregister

Übersicht der Systemregister von GEOS und GEOS/MegaPatch.

Name	Adresse	Größe	Beschreibung
CPU_DDR	\$0000	1 Byte	6510 Datenrichtungsregister
CPU_DATA	\$0001	1 Byte	6510 Datenregister: \$30: RAM_64K \$35: IO_IN \$36: KRNL_IO_IN \$37: KRNL_BAS_IO_IN
GEOS-Register	\$0002	16 Word	Parameterübergabe-Register: r0 = \$0002 r1 = \$0004 r2 = \$0006 r3 = \$0008 r4 = \$000a r5 = \$000c r6 = \$000e r7 = \$0010 r8 = \$0012 r9 = \$0014 r10 = \$0016 r11 = \$0018 r12 = \$001a r13 = \$001c r14 = \$001e r15 = \$0020
curPattern	\$0022	1 Word	Zeiger auf Füllmuster
string	\$0024	1 Word	Zeiger Bereich für Stringeingabe
baselineOffset	\$0026	1 Byte	Abstand bis zur Grundlinie
curSetWidth	\$0027	1 Word	Breite Bitstream-Reihe in Byte
curHeight	\$0029	1 Byte	Anzahl Bitstream-Reihen
curIndexTable	\$002a	1 Word	Zeiger auf Indextabelle
cardDataPntr	\$002c	1 Word	Zeiger auf Bitstream-Grafikdaten
currentMode	\$002e	1 Byte	Schriftartmodus: Bit 0: ungenutzt Bit 1: subscript (tiefstellen) Bit 2: superscript (hochstellen) Bit 3: outline Bit 4: italic (kursiv) Bit 5: invert (revers) Bit 6: bold (fett) Bit 7: underline (unterstrichen)
dispBufferOn	\$002f	1 Byte	Vorder- oder Hintergrundmodus: ST_WR_FORE = \$80: Vordergrund ST_WR_BACK = \$40: Hintergrund
mouseOn	\$0030	1 Byte	Status Mauszeiger: SET_MSE_ON = %10000000 SET_MENUON = %01000000 SET_ICONSON = %00100000
msePicPtr	\$0031	1 Word	Zeiger auf Grafik für Mauszeiger
windowTop	\$0033	1 Byte	Oberer Rand für Textausgaben
windowBottom	\$0034	1 Byte	Unterer Rand für Textausgaben
leftMargin	\$0035	1 Word	Linker Rand für Textausgaben
rightMargin	\$0037	1 Word	Rechter Rand für Textausgaben
pressFlag	\$0039	1 Byte	Tastenstatus: SET_KEYPRESS = %10000000 SET_INPUTCHG = %01000000 SET_MOUSE = %00100000
mouseXPos	\$003a	1 Word	x-Koordinate Mauszeiger
mouseYPos	\$003c	1 Byte	y-Koordinate Mauszeiger

Name	Adresse	Größe	Beschreibung
returnAddress	\$003d	1 Word	Rücksprungadresse Inline-Routine
graphMode	\$003f	1 Byte	Nur GEOS128: Bildschirmflag \$00: 40Z-Modus \$80: 80Z-Modus
GEOS-Register	\$0058- \$006f	24 Byte	Userspace-Register: Der cc65 CrossAssembler verwendet für C-Programme diesen Speicherbereich für diverse Zeiger. Die Verwendung in eigenen Assembler-Programmen sollte damit ebenfalls möglich sein. Von BSW wurden für diesen Bereich keine Labels definiert. Möglich wären folgende Bezeichnungen: u0 = \$0058 u1 = \$005a u2 = \$005c u3 = \$005e u4 = \$0060 u5 = \$0062 u6 = \$0064 u7 = \$0066 u8 = \$0068 u9 = \$006a u10 = \$006c u11 = \$006e
sharedZPage	\$0080- \$00fa	123 Byte	Wird von Anwendungen und einigen Laufwerkstreibern genutzt.
GEOS-Register	\$00fb- \$00fe \$0070- \$007f	2 Word 8 Word	Application-Register: a0 = \$00fb a1 = \$00fd a2 = \$0070 a3 = \$0072 a4 = \$0074 a5 = \$0076 a6 = \$0078 a7 = \$007a a8 = \$007c a9 = \$007e
STATUS	\$0090	1 Byte	Statusbyte für I/O-Operationen
curDevice	\$00ba	1 Byte	Geräteadresse für I/O-Operationen
unused	\$00ff	1 Byte	Nicht verwendet
irqvec	\$0314	1 Word	Kernal-Interruptvektor
bkvec	\$0316	1 Word	Kernal-Break-Vektor
nmivec	\$0318	1 Word	Kernal-NMI-Vektor
kernalVectors	\$031a	10 Word	Zeiger auf Tabelle Kernalroutinen
diskBlkBuf	\$8000	256 Byte	Diskettenblock
fileHeader	\$8100	256 Byte	Infoblock / VLIR-Header
curDirHead	\$8200	256 Byte	Zwischenspeicher BAM
fileTrScTab	\$8300	256 Byte	Track-/Sektor-Tabelle
dirEntryBuf	\$8400	30 Byte	Verzeichniseintrag
DrACurDkNm	\$841e	18 Byte	Diskname Laufwerk A
DrBCurDkNm	\$8430	18 Byte	Diskname Laufwerk B
dataFileName	\$8442	17 Byte	Name Datenfile für GetFile
dataDiskName	\$8453	17 Byte	Name der Disk für GetFile
PrntFilename	\$8465	17 Byte	Aktueller Druckername
PrntDiskName	\$8476	17 Byte	Diskname für Druckertreiber
curDrive	\$8489	1 Byte	Aktuelles Laufwerk 8-11
isGEOS	\$848a	1 Byte	\$ff: Diskette geöffnet
isGEOS	\$848b	1 Byte	\$ff: GEOS-Diskette im Laufwerk
interleave	\$848c	1 Byte	Sektorabstand für TurboDOS
numDrives	\$848d	1 Byte	Anzahl installierter Laufwerke
driveType	\$848e	4 Byte	Laufwerkstyp

Name	Adresse	Größe	Beschreibung
turboFlags	\$8492	4 Byte	\$00: TurboDOS nicht installiert \$80: TurboDOS installiert \$c0: TurboDOS installiert / aktiv
curRecord	\$8496	1 Byte	Aktueller VLIR-Datensatz
usedRecords	\$8497	1 Byte	Anzahl VLIR-Datensätze
fileWritten	\$8498	1 Byte	\$ff = VLIR-Header verändert
fileSize	\$8499	1 Word	Blockgröße einer VLIR-Datei
appMain	\$849b	1 Word	Zeiger Anwender-Mainloop-Routine
intTopVector	\$849d	1 Word	Zeiger Routine zu Beginn des IRQ
intBotVector	\$849f	1 Word	Zeiger Routine am Ende des IRQ
mouseVector	\$84a1	1 Word	Zeiger Routine Mausclick-Abfrage
keyVector	\$84a3	1 Word	Zeiger Routine Tastatur-Abfrage
inputVector	\$84a5	1 Word	Zeiger Routine Ende Texteingabe
mouseFaultVec	\$84a7	1 Word	Zeiger Routine Mausgrenze erreicht
otherPressVec	\$84a9	1 Word	Zeiger Routine für Mausclick außerhalb von Mausgrenzen
StringFaultVec	\$84ab	1 Word	Zeiger Routine für Textausgabe außerhalb von Textgrenzen
alarmTmtVector	\$84ad	1 Word	Zeiger Routine "Alarm ausgelöst"
BRKVector	\$84af	1 Word	Zeiger PANIC!-Routine
RecoverVector	\$84b1	1 Word	Zeiger Routine für Menü-Abbau
selectionFlash	\$84b3	1 Byte	Blinkfrequenz Icons und Menüs
alphaFlag	\$84b4	1 Byte	Blinkfrequenz bei Texteingaben Bit 7=1: Alphanumerische Eingabe Bit 6=1: Cursor ist sichtbar
iconSelFlag	\$84b5	1 Byte	DoIcons: Bit 7=1: Icons blinken Bit 6=1: Icons nur invertieren
faultData	\$84b6	1 Byte	Zeiger hat Mausgrenzen erreicht: Bit 7=1: obere Grenze erreicht Bit 6=1: untere Grenze erreicht Bit 5=1: linke Grenze erreicht Bit 4=1: rechte Grenze erreicht Bit 3=1: Mauspfel nicht auf Menü Die Bit 2-0 sind nicht belegt
menuNumber	\$84b7	1 Byte	Nummer des aktuellen Menüs
mouseTop	\$84b8	1 Byte	y-Koordinate obere Mausgrenze
mouseBottom	\$84b9	1 Byte	y-Koordinate untere Mausgrenze
mouseLeft	\$84ba	1 Word	x-Koordinate linke Mausgrenze
mouseRight	\$84bc	1 Word	x-Koordinate rechte Mausgrenze
stringX	\$84be	1 Word	x-Koordinate für Texteingaben
stringY	\$84c0	1 Byte	y-Koordinate für Texteingaben
mousePicData	\$84c1	64 Byte	Spritedaten für Mauszeiger
maxMouseSpeed	\$8501	1 Byte	Max. Geschwindigkeit Mauszeiger
minMouseSpeed	\$8502	1 Byte	Min. Geschwindigkeit Mauszeiger
mouseAccel	\$8503	1 Byte	Beschleunigungsfaktor Mauszeiger
			minMouseSpeed, maxMouseSpeed und mouseAccel werden vom Joystick-Treiber verwendet um die Zeiger-geschwindigkeit zu kontrollieren

Name	Adresse	Größe	Beschreibung
keyData	\$8504	1 Byte	ASCII-Wert der letzten Taste
mouseData	\$8505	1 Byte	\$00 = Feuerknopf gedrückt \$80 = Feuerknopf nicht gedrückt
inputData	\$8506	4 Byte	Zwischenspeicher Eingabetreiber
inputData +0	\$8506	1 Byte	Aktuelle Richtung Eingabetreiber: 0 = Bewegung nach rechts 1 = Bewegung nach rechts oben 2 = Bewegung nach oben 3 = Bewegung nach links oben 4 = Bewegung nach links 5 = Bewegung nach links unten 6 = Bewegung nach unten 7 = Bewegung nach rechts unten
inputData +1	\$8507	3 Byte	Zwischenspeicher Eingabetreiber Joystick: inputData+0 enthält die aktuelle Zeigergeschwindigkeit
random	\$850a	1 Word	Zufallszahl.
saveFontTab	\$850c	9 Byte	Zwischenspeicher Zeichensatz/Menü
dblClickCount	\$8515	1 Byte	Zähler für Doppelklick-Auswertung
year	\$8516	1 Byte	Jahreszahl (0-99)
month	\$8517	1 Byte	Monat (1-12)
day	\$8518	1 Byte	Tag (1-31)
hour	\$8519	1 Byte	Stunde (0-23)
minutes	\$851a	1 Byte	Minuten (0-59)
seconds	\$851b	1 Byte	Sekunden (0-59) Hinweis: Die Adressen hour, minutes und seconds sind ReadOnly!
alarmSetFlag	\$851c	1 Byte	\$ff: Alarmzeit erreicht
sysDBData	\$851d	1 Byte	Rückmeldung für DoDlgBox
screenColors	\$851e	1 Byte	Vorder- und Hintergrundfarbe
dlgBoxRamBuf	\$851f	417 Byte	Zwischenspeicher für Dialogbox und DeskAccessories
c128_alphaFlag	\$881a	1 Byte	wie alphaFlag, nur 80Z-Modus
DB_DblBit	\$8871	1 Byte	Nur GEOS128: Verdopplung Dialogbox-Icons
savedmoby2	\$88bb	1 Byte	Zwischenspeicher Spritedaten für Dialogbox/DeskAccessory
scr80polar	\$88bc	1 Byte	Nur GEOS128: Kopie des Registers 24 des VDC
scr80colors	\$88bd	1 Byte	Nur GEOS128: Bildschirmfarbe VDC (Register 26)
vdcClrMode	\$88be	1 Byte	Nur GEOS128: Farbmodus VDC
driveData	\$88bf	4 Byte	Zwischenspeicher Laufwerkstreiber
ramExpSize	\$88c3	1 Byte	Größe DAC in 64Kb-Blöcken
sysRAMFlg	\$88c4	1 Byte	GEOS-Systemregister
firstBoot	\$88c5	1 Byte	\$00: Startvorgang aktiv \$ff: Startvorgang beendet
curType	\$88c6	1 Byte	Aktueller Laufwerkstyp
ramBase	\$88c7	4 Byte	Startadresse RAM-Laufwerk in REU
inputDevName	\$88cb	17 Byte	Name des aktuellen Eingabetreiber

Name	Adresse	Größe	Beschreibung
DrCCurDkNm	\$88dc	18 Byte	Diskname Laufwerk C
DrDCurDkNm	\$88ee	18 Byte	Diskname Laufwerk D
dir2Head	\$8900	256 Byte	Zwischenspeicher für BAM
spr0pic	\$8a00	64 Byte	Grafikdaten für Sprite 0 (Maus)
spr1pic	\$8a40	64 Byte	Grafikdaten für Sprite 1 (Cursor)
spr2pic	\$8a80	64 Byte	Grafikdaten für Sprite 2
spr3pic	\$8ac0	64 Byte	Grafikdaten für Sprite 3
spr4pic	\$8b00	64 Byte	Grafikdaten für Sprite 4
spr5pic	\$8b40	64 Byte	Grafikdaten für Sprite 5
spr6pic	\$8b80	64 Byte	Grafikdaten für Sprite 6
spr7pic	\$8bc0	64 Byte	Grafikdaten für Sprite 7
COLOR_MATRIX	\$8c00	1000 Byte	Farben für Vordergrundgrafik
sysApplData	\$8fe8	16 Byte	Zwischenspeicher für Desktop
obj0Pointer	\$8ff8	1 Byte	Zeiger auf Grafik für Sprite 0
obj1Pointer	\$8ff9	1 Byte	Zeiger auf Grafik für Sprite 1
obj2Pointer	\$8ffa	1 Byte	Zeiger auf Grafik für Sprite 2
obj3Pointer	\$8ffb	1 Byte	Zeiger auf Grafik für Sprite 3
obj4Pointer	\$8ffc	1 Byte	Zeiger auf Grafik für Sprite 4
obj5Pointer	\$8ffd	1 Byte	Zeiger auf Grafik für Sprite 5
obj6Pointer	\$8ffe	1 Byte	Zeiger auf Grafik für Sprite 6
obj7Pointer	\$8fff	1 Byte	Zeiger auf Grafik für Sprite 7
DDrvNmData	\$9082	7 Byte	Daten für NativeMode-Disketten
DskDrvBaseL	\$9f7e	4 Byte	Lowbyte Startadresse für Laufwerkstreiber A bis D in REU
DskDrvBaseH	\$9f82	4 Byte	Highbyte Startadresse für Laufwerkstreiber A bis D in REU
doubleSideFlg	\$9f86	4 Byte	\$80: doppelseitige Diskette
drivePartData	\$9f8a	4 Byte	Partition auf CMD-Laufwerken
RealDrvType	\$9f8e	4 Byte	Gerätetyp und Emulationsformat: \$01: C=1541 / SD2IEC-1541 \$02: C=1571 / SD2IEC-1571 \$03: C=1581 / SD2IEC-1581 \$04: SD2IEC- / IECBUS-Native \$05: C=1581 - DOS-Modus \$1x: CMD-FD2000/4000 \$2x: CMD-HD \$3x: CMD-RAMLink \$4x: Shadow-Laufwerk (Cache) \$8x: RAM-Laufwerk \$a4: RAM-Laufwerk C=REU-Native \$b4: RAM-Laufwerk GeoRAM-Native \$c4: RAM-Laufwerk SuperRAM-Native
RealDrvMode	\$9f92	4 Byte	Laufwerkeigenschaften: Bit 7: CMD-Partitionen Bit 6: Native-Verzeichnisse Bit 5: RAM-Laufwerk / CMD-HDPP Bit 4: CMD-SuperRAM-Laufwerk Bit 3: C=REU-Laufwerk Bit 2: GeoRAM-Laufwerk Bit 1: SD2IEC-Laufwerk Bit 0: Nicht verwendet

Name	Adresse	Größe	Beschreibung
RamBankInUse	\$9f96	16 Byte	Speicherbelegungstabelle: %00: nicht belegt %01: Anwendungen %10: Laufwerkstreiber %11: GEOS-System
RamBankFirst	\$9fa6	1 Word	Startadresse DACC in REU
GEOS_RAM_TYP	\$9fa8	1 Byte	Typ Speichererweiterung/REU: \$10: CMD-SuperCPU / RAMCard \$20: GeoRAM / BBGRAM \$40: C=REU \$80: CMD-RAMLink / DACC-Partition
MP3_64K_SYSTEM	\$9fa9	1 Byte	Speicherbank MP3-Kernal
MP3_64K_DATA	\$9faa	1 Byte	Speicherbank MP3-Daten
MP3_64K_DISK	\$9fab	1 Byte	Speicherbank Laufwerkstreiber
Flag_Optimize	\$9fac	1 Byte	Optimierung für CMD-SuperCPU: \$00: GEOS-Optimierung ein \$03: GEOS-Optimierung aus
millenium	\$9fad	1 Byte	Jahrtausend-Byte
Flag_LoadPrnt	\$9fae	1 Byte	MP3-64: \$80: Druckertreiber im RAM
PrntFileNameRAM	\$9faf	17 Byte	MP3-64: Name Druckertreiber im RAM
Flag_Spooler	\$9fc0	1 Byte	Druckerspooler: \$80: Druckerspooler aktiv \$40: Spoolermenü starten
Flag_SpoolMinB	\$9fc1	1 Byte	Erste Speicherbank für Spooler
Flag_SpoolMaxB	\$9fc2	1 Byte	Letzte Speicherbank für Spooler
Flag_SpoolADDR	\$9fc3	3 Byte	Aktuelle Position im Spooler-RAM
Flag_SpoolCount	\$9fc6	1 Byte	Anzahl Dokumente im Spooler
Flag_SplCurDok	\$9fc7	1 Byte	Aktuelles Dokument im Spooler
Flag_SplMaxDok	\$9fc8	1 Byte	Max. Dokumente im Spooler (15)
Flag_TaskAktiv	\$9fc9	1 Byte	TaskManager: \$00: TaskManager aktiv \$ff: TaskManager blockiert
Flag_TaskBank	\$9fca	1 Byte	Systemspeicherbak für TaskManager
Flag_ExtRAMinUse	\$9fcb	1 Byte	GEOS-Speicher belegt: \$80: Hilfsmittel ist geöffnet \$40: Dialogbox ist geöffnet
Flag_ScrSvCnt	\$9fcc	1 Byte	Bildschirmschoner: Aktivierungszeit
Flag_ScrSaver	\$9fcd	1 Byte	Statusbyte für Bildschirmschoner: \$80: deaktiviert \$40: Aktivierungszeit setzen \$20: Aktivierungszeit zählen \$00: Effekt starten
Flag_CrsrRepeat	\$9fce	1 Byte	Verzögerung Tastenwiederholung
BackScrPattern	\$9fcf	1 Byte	Füllmuster für Hintergrundbild
Flag_SetColor	\$9fd0	1 Byte	Farbe in Dialogboxen: \$80: Farbe immer setzen \$40: Nur bei Standard-Dialogbox \$00: Ohne Farbe anzeigen
Flag_ColorDBox	\$9fd1	1 Byte	DoDlgBox: \$ff: Farbe aktiv

Name	Adresse	Größe	Beschreibung
Flag_IconMinX	\$9fd2	1 Byte	DoDlgBox: Mindestbreite für Icons in Farbe
Flag_IconMinY	\$9fd3	1 Byte	DoDlgBox: Mindesthöhe für Icons in Farbe
Flag_IconDown	\$9fd4	1 Byte	DoDlgBox: Grenze für Icons bei Farbe nach unten verschieben
Flag_DBoxType	\$9fd5	1 Byte	DoDlgBox: Kopfbyte, Standard=%1000 0001
Flag_GetFiles	\$9fd6	1 Byte	DoDlgBox: \$80=DBGETFILES \$c0=DBUSRFILES
DB_GFileType	\$9fd7	1 Byte	DBGETFILES: GEOS-Dateityp
DB_GFileClass	\$9fd8	1 Word	DBGETFILES: Zeiger GEOS-Klasse
DB_GetFileEntry	\$9fda	1 Byte	DBUSRFILES: Nr. gewählter Eintrag
DB_StdBoxSize	\$9fdb	6 Byte	DoDlgBox: Standard-Dialogbox Obere y-Koordinate = \$20 Untere y-Koordinate = \$7f Linke x-Koordinate = \$0040 Rechte x-Koordinate = \$00ff
Flag_SetMLine	\$9fe1	1 Byte	Optionen für DoMenu: \$80: Trennlinien anzeigen
Flag_MenuStatus	\$9fe2	1 Byte	Optionen für DoMenu: Bit 7: Menü invertieren Bit 6: nicht nach unten verlassen
DM_LastEntry	\$9fe3	6 Byte	DoMenu: Grafik-Koordinaten Menüeintrag
DM_LastNumEntry	\$9fe9	1 Byte	DoMenu: Nummer invertierter Menüeintrag
Farbtabelle	\$9fea	22 Byte	Farbtabelle GEOS/MegaPatch:
C_Balken	\$9fea	1 Byte	Scrollbar
C_Register	\$9feb	1 Byte	Aktiver Registerkarten-Reiter
C_RegisterOff	\$9fec	1 Byte	Inaktiver Registerkarten-Reiter
C_RegisterBack	\$9fed	1 Byte	Hintergrundfarbe Registerkarten
C_Mouse	\$9fee	1 Byte	Mauszeiger
C_DBoxTitel	\$9fef	1 Byte	Dialogbox-Titelzeile
C_DBoxBack	\$9ff0	1 Byte	Dialogbox-Hintergrund
C_DBoxDIcon	\$9ff1	1 Byte	Dialogbox-Icons
C_FBoxTitel	\$9ff2	1 Byte	Dateiauswahlbox-Titelzeile
C_FBoxBack	\$9ff3	1 Byte	Dateiauswahlbox-Hintergrund
C_FBoxDIcon	\$9ff4	1 Byte	Dateiauswahlbox-Icons
C_FBoxFiles	\$9ff5	1 Byte	Dateiauswahlbox-Dateifenster
C_WinTitel	\$9ff6	1 Byte	Fenster-Titelzeile
C_WinBack	\$9ff7	1 Byte	Fenster-Hintergrund
C_WinShadow	\$9ff8	1 Byte	Fenster-Schatten
C_WinIcon	\$9ff9	1 Byte	Fenster-Icons
C_PullDMenu	\$9ffa	1 Byte	PullDown-Menüs
C_InputField	\$9ffb	1 Byte	Eingabefelder
C_InputFieldOff	\$9ffc	1 Byte	Inaktives Optionsfeld
C_GEOS_BACK	\$9ffd	1 Byte	GEOS-Hintergrundfarbe
C_GEOS_FRAME	\$9ffe	1 Byte	GEOS-Bildschirmrahmenfarbe
C_GEOS_MOUSE	\$9fff	1 Byte	GEOS-Mauszeiger (Kopie C_Mouse)

K.1.3 GEOS-Systemroutinen

Speicherübersicht der Systemroutinen von
GEOS und GEOS/MegaPatch.

Name	Adresse
DoRegister	= \$6d00
ExitRegisterMenu	= \$6d03
RegisterInitMenu	= \$6d06
RegisterUpdate	= \$6d09
RegisterAllOpt	= \$6d0c
RegisterNextOpt	= \$6d0f
RegDrawOptFrame	= \$6d12
RegClrOptFrame	= \$6d15
RegisterSetFont	= \$6d18
InitForPrint	= \$7900
StartPrint	= \$7903
PrintBuffer	= \$7906
StopPrint	= \$7909
GetDimensions	= \$790c
PrintASCII	= \$790f
StartASCII	= \$7912
SetNLQ	= \$7915
PrintFCodes	= \$7918
Get1stDirEntry	= \$9030
GetNxtDirEntry	= \$9033
GetOPDPtr	= \$9036
GetDiskBlkBuf	= \$903c
PutDiskBlkBuf	= \$903f
AllocateBlock	= \$9048
ReadLink	= \$904b
OpenRootDir	= \$9050
OpenSubDir	= \$9053
GetBAMBlock	= \$9056
PutBAMBlock	= \$9059
GetPDirEntry	= \$905c
ReadPDirEntry	= \$905f
OpenPartition	= \$9062
SwapPartition	= \$9065
GetPTypeData	= \$9068
SendFloppyCom	= \$906b
InitForDskDvOp	= \$907c
DoneWithDskDvOp	= \$907f
BootGEOS	= \$c000
ResetHandle	= \$c003
i_UserColor	= \$c0dc
i_ColorBox	= \$c0df
DirectColor	= \$c0e2
RecColorBox	= \$c0e5
GetBackScreen	= \$c0e8
ResetScreen	= \$c0eb
GEOS_InitSystem	= \$c0ee
PutKeyInBuffer	= \$c0f1
SCPU_Pause	= \$c0f4
SCPU_OptOn	= \$c0f7
SCPU_OptOff	= \$c0fa
SCPU_SetOpt	= \$c0fd
InterruptMain	= \$c100
InitProcesses	= \$c103

RestartProcess	= \$c106
EnableProcess	= \$c109
BlockProcess	= \$c10c
UnblockProcess	= \$c10f
FreezeProcess	= \$c112
UnfreezeProcess	= \$c115
HorizontalLine	= \$c118
InvertLine	= \$c11b
RecoverLine	= \$c11e
VerticalLine	= \$c121
Rectangle	= \$c124
FrameRectangle	= \$c127
InvertRectangle	= \$c12a
RecoverRectangle	= \$c12d
DrawLine	= \$c130
DrawPoint	= \$c133
GraphicsString	= \$c136
SetPattern	= \$c139
GetScanLine	= \$c13c
TestPoint	= \$c13f
BitmapUp	= \$c142
PutChar	= \$c145
PutString	= \$c148
UseSystemFont	= \$c14b
StartMouseMode	= \$c14e
DoMenu	= \$c151
RecoverMenu	= \$c154
RecoverAllMenus	= \$c157
DoIcons	= \$c15a
DShiftLeft	= \$c15d
BBMult	= \$c160
BMult	= \$c163
DMult	= \$c166
Ddiv	= \$c169
DSdiv	= \$c16c
Dabs	= \$c16f
Dnegate	= \$c172
Ddec	= \$c175
ClearRam	= \$c178
FillRam	= \$c17b
MoveData	= \$c17e
InitRam	= \$c181
PutDecimal	= \$c184
GetRandom	= \$c187
MouseUp	= \$c18a
MouseOff	= \$c18d
DoPreviousMenu	= \$c190
ReDoMenu	= \$c193
GetSerialNumber	= \$c196
Sleep	= \$c199
ClearMouseMode	= \$c19c
i_Rectangle	= \$c19f
i_FrameRectangle	= \$c1a2
i_RecoverRectangle	= \$c1a5
i_GraphicsString	= \$c1a8
i_BitmapUp	= \$c1ab
i_PutString	= \$c1ae
GetRealSize	= \$c1b1
i_FillRam	= \$c1b4
i_MoveData	= \$c1b7
GetString	= \$c1ba
GotoFirstMenu	= \$c1bd

InitTextPrompt	= \$c1c0	NextRecord	= \$c27a
MainLoop	= \$c1c3	PreviousRecord	= \$c27d
DrawSprite	= \$c1c6	PointRecord	= \$c280
GetCharWidth	= \$c1c9	DeleteRecord	= \$c283
LoadCharSet	= \$c1cc	InsertRecord	= \$c286
PosSprite	= \$c1cf	AppendRecord	= \$c289
EnablSprite	= \$c1d2	ReadRecord	= \$c28c
DisablSprite	= \$c1d5	WriteRecord	= \$c28f
CallRoutine	= \$c1d8	SetNextFree	= \$c292
CalcBlksFree	= \$c1db	UpdateRecordFile	= \$c295
ChkDkGEOS	= \$c1de	GetPtrCurDkNm	= \$c298
NewDisk	= \$c1e1	PromptOn	= \$c29b
GetBlock	= \$c1e4	PromptOff	= \$c29e
PutBlock	= \$c1e7	OpenDisk	= \$c2a1
SetGEOSDisk	= \$c1ea	DoInlineReturn	= \$c2a4
SaveFile	= \$c1ed	GetNextChar	= \$c2a7
SetGDirEntry	= \$c1f0	BitmapClip	= \$c2aa
BldGDirEntry	= \$c1f3	FindBAMBit	= \$c2ad
GetFreeDirBlk	= \$c1f6	SetDevice	= \$c2b0
WriteFile	= \$c1f9	IsMseInRegion	= \$c2b3
BlkAlloc	= \$c1fc	ReadByte	= \$c2b6
ReadFile	= \$c1ff	FreeBlock	= \$c2b9
SmallPutChar	= \$c202	ChangeDiskDevice	= \$c2bc
FollowChain	= \$c205	RstrFrmDialogue	= \$c2bf
GetFile	= \$c208	Panic	= \$c2c2
FindFile	= \$c20b	BitOtherClip	= \$c2c5
CRC	= \$c20e	StashRAM	= \$c2c8
LdFile	= \$c211	FetchRAM	= \$c2cb
EnterTurbo	= \$c214	SwapRAM	= \$c2ce
LdDeskAcc	= \$c217	VerifyRAM	= \$c2d1
ReadBlock	= \$c21a	DoRAMop	= \$c2d4
LdApplc	= \$c21d	TempHideMouse	= \$c2d7
WriteBlock	= \$c220	SetMsePic	= \$c2da
VerWriteBlock	= \$c223	SetNewMode	= \$c2dd
FreeFile	= \$c226	NormalizeX	= \$c2e0
GetFHdrInfo	= \$c229	MoveBData	= \$c2e3
EnterDeskTop	= \$c22c	SwapBData	= \$c2e6
StartAppl	= \$c22f	VerifyBData	= \$c2e9
ExitTurbo	= \$c232	DoB0p	= \$c2ec
PurgeTurbo	= \$c235	AccessCache	= \$c2ef
DeleteFile	= \$c238	HideOnlyMouse	= \$c2f2
FindFTypes	= \$c23b	SetColorMode	= \$c2f5
RstrAppl	= \$c23e	ColorCard	= \$c2f8
ToBasic	= \$c241	ColorRectangle	= \$c2fb
FastDelFile	= \$c244	DoSoftSprites	= \$e045
GetDirHead	= \$c247	InitMouse	= \$fe80
PutDirHead	= \$c24a	SlowMouse	= \$fe83
NxtBlkAlloc	= \$c24d	UpdateMouse	= \$fe86
ImprintRectangle	= \$c250	SetMouse	= \$fe89
i_ImprintRectangle	= \$c253		
DoDlgBox	= \$c256		
RenameFile	= \$c259		
InitForIO	= \$c25c		
DoneWithIO	= \$c25f		
DShiftRight	= \$c262		
CopyString	= \$c265		
CopyFString	= \$c268		
CmpString	= \$c26b		
CmpFString	= \$c26e		
FirstInit	= \$c271		
OpenRecordFile	= \$c274		
CloseRecordFile	= \$c277		

K.1.4 GEOS-Systemroutinen

Alphabetische Übersicht der Systemroutinen von GEOS und GEOS/MegaPatch.

In der Spalte »Seite« verweist die erste Seitenzahl auf die vollständige Beschreibung der Routine, die zweite Seitenzahl verweist auf die Kurzreferenz.

GEOS-Funktion	Key	Adr.	geos64	geos128	MP3-64	MP3-128	Seite
AccessCache	12.3.11	\$c2ef		X		X	302/630
AllocateBlock	12.2.9	\$9048	X	X	X	X	291/617
AppendRecord	12.4.7	\$c289	X	X	X	X	304/636
BBMult	9.1.1	\$c160	X	X	X	X	268/598
BitmapClip	4.23	\$c2aa	X	X	X	X	237/572
BitmapUp	4.21	\$c142	X	X	X	X	236/571
BitOtherClip	4.24	\$c2c5	X	X	X	X	239/572
BldGDirEntry	12.2.15	\$c1f3	X	X	X	X	294/619
BlkAlloc	12.2.7	\$c1fc	X	X	X	X	290/616
BlockProcess	10.4	\$c10c	X	X	X	X	276/602
BMult	9.1.2	\$c163	X	X	X	X	269/598
BootGEOS	14.8	\$c000	X	X	X	X	312/643
CalcBlksFree	12.2.3	\$c1db	X	X	X	X	288/614
CallRoutine	14.4	\$c1d8	X	X	X	X	310/641
ChangeDiskDevice	11.4	\$c2bc	X	X	X	X	280/603
ChangeDiskDevice **)	12.3.1	\$c2bc	X	X	X	X	299/627
ChkDkGEOS	12.2.2	\$c1de	X	X	X	X	287/614
ClearMouseMode	6.4	\$c19c	X	X	X	X	255/586
ClearRam	8.1	\$c178	X	X	X	X	262/592
CloseRecordFile	12.4.2	\$c277	X	X	X	X	303/633
CmpFString	5.12	\$c26e	X	X	X	X	251/583
CmpString	5.11	\$c26b	X	X	X	X	250/583
ColorCard	4.28	\$c2f8		X		X	243/575
ColorRectangle	4.29	\$c2fb		X		X	243/575
CopyFString	5.14	\$c268	X	X	X	X	252/584
CopyString	5.13	\$c265	X	X	X	X	251/583
CRC	14.10	\$c20e	X	X	X	X	313/644
Dabs	9.3.1	\$c16f	X	X	X	X	271/600
Ddec	9.3.3	\$c175	X	X	X	X	271/600
Ddiv	9.2.1	\$c169	X	X	X	X	270/599
DeleteFile	12.1.6	\$c238	X	X	X	X	284/607
DeleteRecord	12.4.9	\$c283	X	X	X	X	305/637
DirectColor	4.33	\$c0e2			X	X	486/576
DisablSprite	7.5	\$c1d5	X	X	X	X	260/591
DMult	9.1.3	\$c166	X	X	X	X	269/598
Dnegate	9.3.2	\$c172	X	X	X	X	271/600
DoBOp	8.8	\$c2ec		X		X	264/594
DoDlgBox	3.1	\$c256	X	X	X	X	215/560
Dolcons	2.2	\$c15a	X	X	X	X	212/559
DoInlineReturn	14.5	\$c2a4	X	X	X	X	311/642
DoMenu	2.1	\$c151	X	X	X	X	204/556

GEOS-Funktion	Key	Adr.	geos64	geos128	MP3-64	MP3-128	Seite
DoneWithDskDvOp	12.3.16	\$907f			X	X	491/632
DoneWithIO	11.2	\$c25f	X	X	X	X	279/603
DoPreviousMenu	2.1.2	\$c190	X	X	X	X	210/557
DoRAMOp	8.11	\$c2d4	X	X	X	X	265/596
DoRegister	15.1	\$6d00			X	X	506/646
DoSoftSprites	7.6	\$e045		X		X	260/591
DrawLine	4.7	\$c130	X	X	X	X	227/565
DrawPoint	4.1	\$c133	X	X	X	X	224/563
DrawSprite	7.2	\$c1c6	X	X	X	X	260/590
DSdiv	9.2.2	\$c16c	X	X	X	X	270/599
DShiftLeft	9.3.4	\$c15d	X	X	X	X	271/600
DShiftRight	9.3.5	\$c262	X	X	X	X	272/600
EnableProcess	10.3	\$c109	X	X	X	X	275/601
EnablSprite	7.4	\$c1d2	X	X	X	X	260/591
EnterDeskTop	14.6	\$c22c	X	X	X	X	311/642
EnterTurbo	11.5	\$c214	X	X	X	X	280/604
ExitRegisterMenu	15.2	\$6d03			X	X	506/646
ExitTurbo	11.6	\$c232	X	X	X	X	281/604
FastDelFile	12.2.20	\$c244	X	X	X	X	296/621
FetchRAM	8.13	\$c2cb	X	X	X	X	266/596
FillRam	8.2	\$c17b	X	X	X	X	262/592
FindBAMBit	12.2.6	\$c2ad	X	X	X	X	289/615
FindFile	12.1.5	\$c20b	X	X	X	X	284/607
FindFTypes	12.1.4	\$c23b	X	X	X	X	283/606
FirstInit	11.8	\$c271	X	X	X	X	281/604
FollowChain	12.2.19	\$c205	X	X	X	X	295/621
FrameRectangle	4.10	\$c127	X	X	X	X	229/566
FreeBlock	12.2.11	\$c2b9	X	X	X	X	292/617
FreeFile	12.2.12	\$c226	X	X	X	X	292/618
FreezeProcess	10.6	\$c112	X	X	X	X	276/602
GEOS_InitSystem	14.12	\$c0ee			X	X	492/644
Get1stDirEntry	12.2.16	\$9030	X	X	X	X	294/620
GetBackScreen	4.34	\$c0e8			X	X	487/577
GetBAMBlock	12.3.12	\$9056			X	X	490/631
GetBlock	12.3.3	\$c1e4	X	X	X	X	300/627
GetDiskBlkBuf	12.3.8	\$903c	X	X	X	X	302/629
GetCharWidth	5.9	\$c1c9	X	X	X	X	250/582
GetDimensions	13.5	\$790c	X	X	X	X	308/639
GetDirHead	12.2.4	\$c247	X	X	X	X	288/615
GetFHdrlInfo	12.2.18	\$c229	X	X	X	X	295/620
GetFile	12.1.8	\$c208	X	X	X	X	284/609
GetFreeDirBlk	12.2.13	\$c1f6	X	X	X	X	293/618
GetNextChar	5.7	\$c2a7	X	X	X	X	249/581
GetNxtDirEntry	12.2.17	\$9033	X	X	X	X	295/620
GetOPDPtr	12.3.10	\$9036	X	X	X	X	302/630
GetPDirEntry	12.1.14	\$905c			X	X	488/613
GetPtrCurDkNm	12.1.2	\$c298	X	X	X	X	282/605
GetPTypeData	12.1.15	\$9068			X	X	489/613

GEOS-Funktion	Key	Adr.	geos64	geos128	MP3-64	MP3-128	Seite
GetRandom	14.3	\$c187	X	X	X	X	310/641
GetRealSize	5.10	\$c1b1	X	X	X	X	250/582
GetScanLine	4.8	\$c13c	X	X	X	X	227/565
GetSerialNumber	14.11	\$c196	X	X	X	X	313/644
GetString	5.6	\$c1ba	X	X	X	X	248/581
GotoFirstMenu	2.1.1	\$c1bd	X	X	X	X	210/557
GraphicsString	4.19	\$c136	X	X	X	X	233/570
HideOnlyMouse	6.11	\$c2f2		X		X	257/588
HorizontalLine	4.3	\$c118	X	X	X	X	226/563
i_BitmapUp	4.22	\$c1ab	X	X	X	X	236/571
i_ColorBox	4.31	\$c0df			X	X	485/576
i_FillRam	8.3	\$c1b4	X	X	X	X	262/592
i_FrameRectangle	4.11	\$c1a2	X	X	X	X	230/566
i_GraphicsString	4.20	\$c1a8	X	X	X	X	236/571
i_ImprintRectangle	4.16	\$c253	X	X	X	X	233/568
i_MoveData	8.6	\$c1b7	X	X	X	X	264/593
i_PutString	5.5	\$c1ae	X	X	X	X	248/580
i_RecoverRectangle	4.18	\$c1a5	X	X	X	X	233/569
i_Rectangle	4.13	\$c19f	X	X	X	X	231/567
i_UserColor	4.32	\$c0dc			X	X	486/576
ImprintRectangle	4.15	\$c250	X	X	X	X	232/568
InitForDskDvOp	12.3.15	\$907c			X	X	491/632
InitForIO	11.1	\$c25c	X	X	X	X	279/603
InitForPrint	13.1	\$7900	X	X	X	X	306/639
InitMouse	6.6	\$fe80	X	X	X	X	255/587
InitProcesses	10.1	\$c103	X	X	X	X	274/601
InitRam	8.4	\$c181	X	X	X	X	263/592
InitTextPrompt	5.18	\$c1c0	X	X	X	X	253/585
InsertRecord	12.4.8	\$c286	X	X	X	X	304/636
InterruptMain	14.2	\$c100	X	X	X	X	310/641
InvertLine	4.5	\$c11b	X	X	X	X	226/564
InvertRectangle	4.14	\$c12a	X	X	X	X	232/568
IsMselInRegion	6.12	\$c2b3	X	X	X	X	258/589
LdApplic	12.2.21	\$c21d	X	X	X	X	296/622
LdDeskAcc	12.2.23	\$c217	X	X	X	X	297/623
LdFile	12.2.24	\$c211	X	X	X	X	297/624
LoadCharSet	5.17	\$c1cc	X	X	X	X	253/585
MainLoop	14.1	\$c1c3	X	X	X	X	310/641
MouseOff	6.3	\$c18d	X	X	X	X	255/586
MouseUp	6.2	\$c18a	X	X	X	X	255/586
MoveBData	8.7	\$c2e3		X		X	264/594
MoveData	8.5	\$c17e	X	X	X	X	263/593
NewDisk	12.2.1	\$c1e1	X	X	X	X	287/614
NextRecord	12.4.5	\$c27a	X	X	X	X	304/635
NormalizeX	4.25	\$c2e0		X		X	242/573
NxtBlkAlloc	12.2.8	\$c24d	X	X	X	X	290/616
OpenDisk	12.1.3	\$c2a1	X	X	X	X	283/606
OpenPartition	12.1.13	\$9062			X	X	488/612

GEOS-Funktion	Key	Adr.	geos64	geos128	MP3-64	MP3-128	Seite
OpenRecordFile	12.4.1	\$c274	X	X	X	X	303/633
OpenRootDir	12.1.11	\$9050			X	X	488/612
OpenSubDir	12.1.12	\$9053			X	X	488/612
Panic	14.7	\$c2c2	X	X	X	X	311/643
PointRecord	12.4.4	\$c280	X	X	X	X	303/634
PosSprite	7.3	\$c1cf	X	X	X	X	260/590
PreviousRecord	12.4.6	\$c27d	X	X	X	X	304/635
PrintASCII	13.7	\$790f	X	X	X	X	308/640
PrintBuffer	13.3	\$7906	X	X	X	X	307/639
PrintFCodes	13.9	\$7918			X	X	309/640
PromptOff	5.16	\$c29e	X	X	X	X	252/584
PromptOn	5.15	\$c29b	X	X	X	X	252/584
PurgeTurbo	11.7	\$c235	X	X	X	X	281/604
PutBAMBlock	12.3.13	\$9059			X	X	490/631
PutBlock	12.3.5	\$c1e7	X	X	X	X	301/628
PutChar	5.1	\$c145	X	X	X	X	244/578
PutDecimal	5.3	\$c184	X	X	X	X	246/579
PutDirHead	12.2.5	\$c24a	X	X	X	X	288/615
PutDiskBlkBuf	12.3.9	\$903f	X	X	X	X	302/629
PutKeyInBuffer	14.13	\$c0f1			X	X	492/645
PutString	5.4	\$c148	X	X	X	X	247/580
ReadBlock	12.3.4	\$c21a	X	X	X	X	300/628
ReadByte	12.2.27	\$c2b6	X	X	X	X	298/625
ReadFile	12.2.25	\$c1ff	X	X	X	X	298/624
ReadLink	12.3.2	\$904b	X	X	X	X	299/627
ReadPDirEntry	12.2.28	\$905f			X	X	490/626
ReadRecord	12.4.10	\$c28c	X	X	X	X	305/637
RecColorBox	4.30	\$c0e5			X	X	485/576
RecoverAllMenus	2.1.5	\$c157	X	X	X	X	211/558
RecoverLine	4.4	\$c11e	X	X	X	X	226/564
RecoverMenu	2.1.4	\$c154	X	X	X	X	211/558
RecoverRectangle	4.17	\$c12d	X	X	X	X	233/569
Rectangle	4.12	\$c124	X	X	X	X	231/567
ReDoMenu	2.1.3	\$c193	X	X	X	X	211/557
RegClrOptFrame	15.8	\$6d15			X	X	507/647
RegDrawOptFrame	15.7	\$6d12			X	X	507/647
RegisterAllOpt	15.5	\$6d0c			X	X	507/646
RegisterInitMenu	15.3	\$6d06			X	X	506/646
RegisterNextOpt	15.6	\$6d0f			X	X	507/646
RegisterSetFont	15.9	\$6d18			X	X	507/647
RegisterUpdate	15.4	\$6d09			X	X	506/646
RenameFile	12.1.7	\$c259	X	X	X	X	284/608
ResetHandle *)		\$c003	X	X	X	X	
ResetScreen	4.35	\$c0eb			X	X	487/577
RestartProcess	10.2	\$c106	X	X	X	X	275/601
RstrAppl	12.1.10	\$c23e	X	X	X	X	287/611
RstrFrmDialogue	3.2	\$c2bf	X	X	X	X	221/562
SaveFile	12.1.9	\$c1ed	X	X	X	X	286/610

GEOS-Funktion	Key	Adr.	geos64	geos128	MP3-64	MP3-128	Seite
SCPU_OptOff	14.16	\$c0fa			X	X	493/645
SCPU_OptOn	14.15	\$c0f7			X	X	493/645
SCPU_Pause	14.14	\$c0f4			X	X	493/645
SCPU_SetOpt	14.17	\$c0fd			X	X	493/645
SendFloppyCom	12.3.14	\$906b			X	X	491/632
SetColorMode	4.27	\$c2f5		X		X	243/574
SetDevice	11.3	\$c2b0	X	X	X	X	280/603
SetGDirEntry	12.2.14	\$c1f0	X	X	X	X	293/619
SetGEOSDisk	12.1.1	\$c1ea	X	X	X	X	282/605
SetMouse	6.8	\$fe89		X		X	256/588
SetMsePic	6.9	\$c2da		X		X	256/588
SetNewMode	4.26	\$c2dd		X		X	243/574
SetNextFree	12.2.10	\$c292	X	X	X	X	291/617
SetNLQ	13.8	\$7915	X	X	X	X	309/640
SetPattern	4.9	\$c139	X	X	X	X	228/566
Sleep	10.8	\$c199	X	X	X	X	278/602
SlowMouse	6.5	\$fe83	X	X	X	X	255/587
SmallPutChar	5.2	\$c202	X	X	X	X	246/579
StartAppl	12.2.22	\$c22f	X	X	X	X	297/623
StartASCII	13.6	\$7912	X	X	X	X	308/640
StartMouseMode	6.1	\$c14e	X	X	X	X	254/586
StartPrint	13.2	\$7903	X	X	X	X	307/639
StashRAM	8.12	\$c2c8	X	X	X	X	266/596
StopPrint	13.4	\$7909	X	X	X	X	307/639
SwapBData	8.9	\$c2e6		X		X	265/595
SwapPartition	12.2.29	\$9065			X	X	490/626
SwapRAM	8.14	\$c2ce	X	X	X	X	266/597
TempHideMouse	6.10	\$c2d7		X		X	257/588
TestPoint	4.2	\$c13f	X	X	X	X	225/563
ToBasic	14.9	\$c241	X	X	X	X	312/644
UnblockProcess	10.5	\$c10f	X	X	X	X	276/602
UnfreezeProcess	10.7	\$c115	X	X	X	X	277/602
UpdateMouse	6.7	\$fe86	X	X	X	X	255/587
UpdateRecordFile	12.4.3	\$c295	X	X	X	X	303/634
UseSystemFont	5.8	\$c14b	X	X	X	X	250/582
VerifyBData	8.10	\$c2e9		X		X	265/595
VerifyRAM	8.15	\$c2d1	X	X	X	X	266/597
VerticalLine	4.6	\$c121	X	X	X	X	227/564
VerWriteBlock	12.3.7	\$c223	X	X	X	X	301/629
WriteBlock	12.3.6	\$c220	X	X	X	X	301/628
WriteFile	12.2.26	\$c1f9	X	X	X	X	298/625
WriteRecord	12.4.11	\$c28f	X	X	X	X	305/638

*) Die Adresse *ResetHandle* ist nur der Vollständigkeit halber aufgeführt. Diese wird von GEOS V2 intern beim Boot-Vorgang aufgerufen um das System zu initialisieren. Für Programme nicht einsetzbar, da die eigentliche Routine nicht im Kernal enthalten ist.

**) Die Routine *ChangeDiskDevice* wird im Original-Handbuch zum MegaAssembler doppelt aufgeführt und ist daher in dieser Tabelle ebenfalls doppelt aufgelistet.

K.2 Menüroutinen

GEOS-Routinen für die Menüsteuerung.

K.2.1 DoMenu (\$c151)

Erstellt ein horizontales oder vertikales PullDown-Menü.

Übergabe:	r0	Zeiger auf Menütabelle.
	a	Zeiger auf Menüeintrag.
Rückgabe:	n/a	
Verwendet:	Flag_SetMLine	Nur MP3: Bit 7=1: Trennlinien im Menü zeichnen.
	Flag_MenuStatus	Nur MP3: Bit 7=1: Eintrag unter Mauszeiger invertieren. Nur MP3: Bit 6=1: Menü nicht nach unten verlassen.
Verändert:	a, x, y, r0 bis r15	StartMouseMode kann r0 bis r15 verändern, daher gilt das auch für DoMenu.
	mouseOn	Bit 5=1: Icon-Menü aktivieren wenn Bit 7=1. Bit 6=1: PullDown-Menü aktivieren. Bit 7=1: Mauszeiger aktiv (über StartMouseMode).
	faultData	\$00: Initialisierung.
	menuNumber	\$00: Hauptmenü.
Aufruf von:	StartMouseMode	Mausabfrage starten.
Hinweis:		Setzen des Mauszeigers auf einen Menüeintrag funktioniert unter GEOS V2 auf Grund eines Fehlers nur bis zu einer y-Koordinate < 128.

Aufbau der Menütabelle:

```

        b yoben, yunten
        w xlinks, xrechts          ; GEOS128: !DOUBLE_W!ADD1_W möglich
                                   ; GEOS V2: x-Koordinate =< 255!

        b Anzahl !VERTICAL !HORIZONTAL! CONSTRAINED !UN_CONSTRAINED

::00h      w Actiontext
           b MENU_ACTION
           w ActionRoutine          ; Zeiger auf Programmroutine.

::80h      w SubText
           b SUB_MENU
           w SubTabelle            ; Zeiger auf Menütabelle.

::40h      w DynSubText
           b DYN_SUB_MENU
           w DynSubRoutine         ; Zeiger auf Menüroutine.

:DynSubRoutine ...

        LoadW r0,SUB_MENU          ; Zeiger auf Submenu übergeben.
        rts

```


K.2.1.1 GotoFirstMenu (\$c1bd)

Hauptmenü aktivieren.

Übergabe:	n/a	
Rückgabe:	n/a	
Verändert:	a, x, y, r0 bis r15 mouseOn	Bit 6=1: PullDown-Menü aktivieren. Bit 7=1: Mauszeiger aktiv (über StartMouseMode). \$00: Initialisierung. \$00: Hauptmenü.
	faultData menuNumber	
Aufruf von:	MouseOff DoPreviousMenu StartMouseMode	Mauszeiger abschalten. Vorheriges Menü aufrufen wenn menuNumber > 0. Mausabfrage starten.

K.2.1.2 DoPreviousMenu (\$c190)

Übergeordnetes Menü aktivieren.

Übergabe:	n/a	
Rückgabe:	n/a	
Verändert:	a, x, y, r0 bis r15 mouseOn	Bit 6=1: PullDown-Menü aktivieren. Bit 7=1: Mauszeiger aktiv (über StartMouseMode). \$00: Initialisierung.
	faultData menuNumber	
Aufruf von:	MouseOff StartMouseMode	Mauszeiger abschalten. Mausabfrage starten.
Hinweis:		Aufruf aus dem Hauptmenü führt zum Absturz.

K.2.1.3 ReDoMenu (\$c193)

Aktuelles Menü erneut anzeigen.

Übergabe:	n/a	
Rückgabe:	n/a	
Verändert:	a, x, y, r0 bis r15 mouseOn	Bit 6=1: PullDown-Menü aktivieren. Bit 7=1: Mauszeiger aktiv (über StartMouseMode). \$00: Initialisierung.
	faultData	
Aufruf von:	MouseOff StartMouseMode	Mauszeiger abschalten Mausabfrage starten

K.2.1.4 RecoverMenu (\$c193)

Aktuelles Menü vom Bildschirm löschen.

Übergabe: n/a

Rückgabe: n/a

Verwendet: RecoverVector Zeiger auf Routine zum wiederherstellen eines Bildschirmbereiches. Zeigt normalerweise auf die Routine RecoverRectangle.
Wenn in RecoverVector=\$0000 steht, dann wird der Bildschirm durch ein leeres Rechteck gelöscht.

Verändert: a, x, y, r0 bis r15

Verwendet von: RecoverAllMenus Alle Menüs vom Bildschirm löschen.

K.2.1.5 RecoverAllMenus (\$c157)

Alle Menüs vom Bildschirm löschen und Menü-Abfrage abschalten.

Übergabe: n/a

Rückgabe: n/a

Verwendet: RecoverVector Zeiger auf Routine zum wiederherstellen eines Bildschirmbereiches. Zeigt normalerweise auf die Routine RecoverRectangle.
Wenn in RecoverVector=\$0000 steht, dann wird der Bildschirm durch ein leeres Rechteck gelöscht.

Verändert: a, x, y, r0 bis r15
mouseOn Bit 6=0: PullDown-Menüs abschalten.
menuNumber \$00: Hauptmenü.

Aufruf von: RecoverMenu Aktuelles Menü vom Bildschirm löschen.

K.2.2 Dolcons (\$c15a)

Erstellt ein Icon-Menü.

Übergabe:	r0	Zeiger auf Menütabelle.
Rückgabe:	n/a	
Verwendet:	iconSelfFlag	Bit 7=1: Icon blinkt bei Auswahl. Bit 6=1: Icon wird bei Auswahl invertiert.
Verändert:	a, x, y, r0 bis r15 mouseOn	Bit 5=1: Icon-Menü aktivieren. Bit 6=0: PullDown-Menü abschalten wenn Bit 7=0. Bit 7=1: Mauszeiger aktiv (über StartMouseMode). \$00: Initialisierung.
	faultData	
Aufruf von:	StartMouseMode	Mausabfrage starten.
Hinweis:		Unter GEOS V1.0-V1.2 muss mindestens ein Icon über Dolcons definiert werden, falls kein Icon-Menü benötigt wird muss ein Dummy-Icon installiert werden. Ab GEOS V1.3 nicht mehr erforderlich.

Aufbau der Menütabelle:

```

b Anzahl                ; Anzahl Icons (max. 31)

w mouseXPos             ; $0000 = Mausposition nicht ändern
b mouseYPos

w iconGraphic            ; Zeiger auf Icon-Grafik
b iconXPos              ; x-Koordinate in Cards
b iconYPos              ; y-Koordinate in Pixel
b iconWidth             ; Breite des Icon in Cards
b iconHeight            ; Höhe des Icon in Pixel
w iconRoutine           ; Zeiger auf Programmroutine

```

Für GEOS V1.0 bis V1.2 muss eine Dummy-Icon-Tabelle an *Dolcons* übergeben werden, auch wenn kein Icon-Menü verwendet wird.

```

b 1                    ; Mind. ein Eintrag

w $0000               ; Mausposition nicht ändern
b $00

w $0000              ; Keine Icon-Grafik
b 0                  ; Dummy x-Koordinate der Grafik
b 0                  ; Dummy y-Koordinate der Grafik
b 1                  ; Dummy-Breite muss > 0 sein!
b 1                  ; Dummy-Höhe muss > 0 sein!
w $0000              ; Keine Programmroutine

```

Ab GEOS V1.3 wird das Highbyte des Zeigers auf eine Menütabelle gelöscht. Mainloop prüft ob das Highbyte 0 ist: In dem Fall wird, auch bei *mouseOn* mit Bit 5=1 die Menütabelle nicht mehr ausgewertet.

Siehe Kernal-Routine *GEOS_Init* und Tabelle ab *InitVarData*.

K.3 Dialogboxroutinen

GEOS-Routinen für Dialogboxen.

K.3.1 DoDlgBox (\$c256)

Öffnet eine Dialogbox auf dem Bildschirm.

Übergabe:	r0	Zeiger auf Dialogboxtabelle.
Rückgabe:	r0L / sysDBData	Status-Byte, Nur MP3: \$88-\$8b für Laufwerk-Icons.
Verändert:	a, x, y, r0 bis r11 r0 bis r15	"Official GEOS Programmers Reference Guide". "The Hitchhikers Guide to GEOS".
Aufruf von:	StartMouseMode	Mausabfrage starten.
Hinweis:		Offset bezieht sich auf die linke, max. 0-255 möglich. Max. können acht Icons definiert werden.

Aufbau der Dialogboxtabelle:

	b TYPE	; Bit 7=1: Standard-Dialogbox ; Bit 6=1: MP3: Erweiterung Farbe ; Bit 5=1: MP3: Ohne Farbe anzeigen
	b yoben, yunten	; Nur wenn Bit 7=0!
	w xlinks, xrechts	; GEOS128: !DOUBLE_W!ADD1_W möglich
	b Farbe	; Nur MP3 mit TYPE/Bit 6=1
::01_06h	b OK / CANCEL / YES / NO / OPEN / DISK	
	b xOffset	; x-Offset in Cards
	b yOffset	; y-Offset in Pixel
	b Farbe	; Nur MP3: TYPE/Bit 6=1
::07h	b DRIVE	; Nur MP3: Laufwerk-Icons ; sysDBData: \$88-\$8d = Laufwerk A-D
	b xOffset	; x-Offset in Cards
	b yOffset	; y-Offset in Pixel
	b Farbe	; Nur MP3: TYPE/Bit 6=1
::08h	b DUMMY	; Nur MP3: Füllbyte (ohne Funktion)
::09h	b DBUSRFILES	; Nur MP3: Listenauswahlbox
	w ListData	; Word, Zeiger auf Liste. ; Alle Einträge 16Z + NULL-Byte!
::0Ah	b DBSETCOL	; Nur MP3: Farbrechteck zeichnen
	b xOffset	; x-Offset in Cards
	b yOffset	; y-Offset in Cards
	b Breite	; Breite in Cards
	b Höhe	; Höhe in Cards
	b Farbe	; Nur MP3 mit TYPE/Bit 6=1
::0Bh	b DBTXTSTR	; Textausgabe
	b xOffset	; x-Offset in Pixel (max. 0-255)
	b yOffset	; y-Offset in Pixel
	w DBoxText	; Zeiger auf Dialogbox-Text

::0Ch	b DBVARSTR b xOffset b yOffset b rXL	; Textausgabe ; x-Offset in Pixel (max. 0-255) ; y-Offset in Pixel ; Lowbyte Adresse r5 bis r10, bei ; Applications auch a0 bis a9, mit ; Zeiger auf Dialogbox-Text
::0Dh	b DBGETSTRING b xOffset b yOffset b rXL b Anzahl	; Texteingabe ; x-Offset in Pixel (max. 0-255) ; y-Offset in Pixel, der Wert wird ; ohne baselineOffset angegeben! ; Lowbyte Adresse r5 bis r10, bei ; Applications auch a0 bis a9, mit ; Zeiger auf Dialogbox-Text ; Anzahl Zeichen für Texteingabe
::0Eh	b DBSYSOPV	; Ende Dialogbox mit Mausklick
::0Fh	b DBGRPHSTR w GraphStrData	; Aufruf von GraphicsString ; Word, Zeiger auf Grafikdaten
::10h	b DBGETFILES !DBSELECTPART !DBSETDRVICON b xOffset b yOffset	; Dateiauswahlbox ; Nur MP3: Partitionsauswahl ; Erfordert das OPEN-Icon! ; Nur MP3: Laufwerk-Icons anzeigen ; Übergabe wie bei DRIVE ; x-Offset in Pixel (max. 0-255) ; Nur MP3: \$00, da ohne Funktion ; y-Offset in Pixel ; Nur MP3: \$00, da ohne Funktion ; r5 =Zeiger auf Ablagebereich ; r7L=GEOS-Filetyp ; Nur MP3: r7L=255: alle Dateitypen ; r10=GEOS-Klasse
::11h	b DBOPVEC w MouseCheck	; Mausklick außerhalb von Icons ; Routine für Mausabfrage
::12h	b DBUSRICON b xOffset b yOffset b Farbe w IconEntry	; Anwender-Icon ; x-Offset in Cards ; y-Offset in Pixel ; Nur MP3 mit TYPE/Bit 6=1 ; Word, Zeiger auf Icon-Eintrag
::13h	DB_USR_ROUT w UserRout	; Anwender-Routine aufrufen ; Word, Zeiger auf Anwender-Routine

Aufbau Eintrag für Anwender-Icon (DBUSRICON):

:IconEntry	w iconGrafik b xOffset b yOffset b iconWidth b iconHeight w iconRoutine	; Word, Zeiger auf Icon-Grafik ; x-Offset in Cards ; y-Offset in Pixel ; Breite des Icon in Cards ; Höhe des Icon in Pixel ; Zeiger auf Programmroutine
------------	--	--

K.3.2 RstrFrmDialogue (\$c2bf)

Dialogbox beenden.

Übergabe:	n/a	Zeiger auf Dialogboxtabelle.
Rückgabe:	r0L	Status-Byte für Hauptanwendung.
Verwendet:	sysDBData	Status-Byte der Dialogbox.
Verändert:	a, x, y, r0 bis r15	
Hinweis:		Das Programm wird hinter dem ursprünglichen Aufruf von DoDlgBox fortgesetzt.

Dialogbox über die Mainloop beenden (Empfohlen):

```

:IconRout    ...                                ; Routine für Mausklick auf ein
                                                    ; Anwender-Icon.

            lda    #STATUS                      ; Status-Byte für Rückmeldung
            sta    sysDBData                    ; für "Mausklick auf Anwender-Icon"

            LoadW appMain,RstrFrmDialogue
            rts                                  ; Zurück zur Hauptanwendung

```

Dialogbox direkt beenden:

```

:IconRout    ...                                ; Routine für Mausklick auf ein
                                                    ; Anwender-Icon.

            lda    #STATUS                      ; Status-Byte für Rückmeldung
            sta    sysDBData                    ; für "Mausklick auf Anwender-Icon"

            jmp    RstrFrmDialogue              ; Zurück zur Hauptanwendung

```

K.3.3 Dialogboxen unter GEOS128

Im 80-Zeichen-Modus sollte die linke x-Koordinate immer mit *DOUBLE_W* definiert werden. Damit wird bei System-Icons die Breite automatisch verdoppelt. Ansonsten zeigt die Dialogbox System-Icons nur mit halber Breite an.

Bei einer Dialogbox ohne Schatten wird das *DOUBLE_W* in der linken x-Koordinate unter GEOS128 nicht ausgewertet, was dazu führt das System-Icons in der Breite nicht mehr verdoppelt werden.

Das Problem lässt sich umgehen wenn man über eine Benutzeroutine mit *DB_USR_ROUT* am Anfang der Dialogboxtabelle eine Routine aufruft, die das *DB_DblBit* (\$003f) auf den Wert \$80 setzt.

Das Problem ist in aktuellen Versionen von GEOS/MegaPatch behoben.

Grundsätzlich gilt:

DOUBLE_W erzeugt gerade x-Koordinaten als (Word, z.B. linker Rand)

DOUBLE_W ! ADD1_W erzeugt ungerade x-Koordinaten (Word, z.B. rechter Rand)

DOUBLE_B erzeugt immer einen geraden x-Offset (Byte, z.B. Icon-Position)

Dialogboxen mit Verdopplungstechnik dürfen nicht unter GEOS64 benutzt werden!

Für weitere Informationen siehe **Teil B, Kapitel 3.3 ab Seite 222**.

K.4 Grafikroutinen

GEOS-Routinen für die Grafikdarstellung. Nur GEOS128: x-Koordinaten können mit *DOUBLE_W*, *ADD1_W* oder *DOUBLE_B* automatisch verdoppelt werden, die übergebenen Parameter in den Registern werden dann direkt verändert.

K.4.1 DrawPoint (\$c133)

Zeichnet einen Punkt im Vorder- oder Hintergrundbildschirm, oder stellt dessen Inhalt aus dem Hintergrundbildschirm wieder her.

Übergabe:	r3	Word, x-Koordinate für Grafikpunkt.
	r11L	Byte, y-Koordinate für Grafikpunkt.
	N-Flag	N=1: Punkt aus Hintergrundbildschirm kopieren. N=0: Punkt in Vorder- und/oder Hintergrund schreiben.
	C-Flag	C=1: Punkt setzen. C=0: Punkt löschen.
Verwendet:	dispBufferOn	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich.
Verändert:	a, x, y, r5, r6	
Unverändert:	r3, r11L	

K.4.2 TestPoint (\$c13f)

Punkt im Vorder- oder Hintergrundbildschirm testen.

Übergabe:	r3	Word, x-Koordinate für Grafikpunkt.
	r11L	Byte, y-Koordinate für Grafikpunkt.
Rückgabe:	C-Flag	C=1: Punkt gesetzt. C=0: Punkt gelöscht.
Verwendet:	dispBufferOn	Bit 7=1: Vordergrund testen. Bit 6=1: Hintergrund testen. Kombination von Bit 7 und Bit 6 möglich.
Verändert:	a, x, y, r5, r6	
Unverändert:	r3, r11L	

K.4.3 HorizontalLine (\$c118)

Zeichnet eine Linie mit einem vorgegebenen Muster.

Übergabe:	r3	Word, x-Koordinate, linkes Ende der Linie.
	r4	Word, x-Koordinate, rechtes Ende der Linie.
	r11L	Byte, y-Koordinate der Linie.
	a	Byte, die 8 Bit definieren das Linienmuster.
Verwendet:	dispBufferOn	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich.
Verändert:	a, x, y r5 bis r8, r11H	
Unverändert:	r3, r4, r11L	

K.4.4 RecoverLine (\$c11e)

Überträgt eine Linie aus dem Hintergrundbildschirm in den Vordergrundbildschirm.

Übergabe:	r3	Word, x-Koordinate, linkes Ende der Linie.
	r4	Word, x-Koordinate, rechtes Ende der Linie.
	r11L	Byte, y-Koordinate der Linie.

Verändert: a, x, y
r5 bis r8

Unverändert: r3, r4, r11L

K.4.5 InvertLine (\$c11b)

Invertiert eine Linie im Vorder- und/oder Hintergrundbildschirm.

Übergabe:	r3	Word, x-Koordinate, linkes Ende der Linie.
	r4	Word, x-Koordinate, rechtes Ende der Linie.
	r11L	Byte, y-Koordinate der Linie.

Verwendet: dispBufferOn Bit 7=1: Linie im Vordergrund invertieren.
Bit 6=1: Linie im Hintergrund invertieren.
Kombination von Bit 7 und Bit 6 möglich.

Verändert: a, x, y
r5 bis r8

Unverändert: r3, r4, r11L

K.4.6 VerticalLine (\$c121)

Zeichnet eine vertikale Linie im Vorder- und/oder Hintergrundbildschirm.

Übergabe:	r4	Word, x-Koordinate der Linie.
	r3L	Byte, y-Koordinate, oberes Ende der Linie.
	r3H	Byte, y-Koordinate, unteres Ende der Linie.

a Byte, die 8 Bit definieren das Linienmuster.

Verwendet: dispBufferOn Bit 7=1: In Vordergrund schreiben.
Bit 6=1: In Hintergrund schreiben.
Kombination von Bit 7 und Bit 6 möglich.

Verändert: a, x, y

r5 bis r7, r8L MegaAssembler-Handbuch.

r5 bis r8	"Official GEOS Programmers Reference Guide" und "The Hitchhikers Guide to GEOS".
-----------	--

Unverändert: r3L, r3H, r4

K.4.7 DrawLine (\$c130)

Zeichnet eine Linie im Vorder- und/oder Hintergrundbildschirm oder überträgt eine Linie aus dem Hintergrund- in den Vordergrundbildschirm.

Übergabe:	r3	Word, x-Koordinate, linkes Ende der Linie.
	r4	Word, x-Koordinate, rechtes Ende der Linie.
	r11L	Byte, y-Koordinate, oberes Ende der Linie.
	r11H	Byte, y-Koordinate, unteres Ende der Linie.
	N-Flag	N=1: Linie aus Hintergrundbildschirm kopieren. N=0: Linie in Vorder- und/oder Hintergrund schreiben.
	C-Flag	C=1: Linie setzen. C=0: Linie löschen.
Verwendet:	dispBufferOn	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich.
Verändert:	a, x, y r3 bis r13	
Hinweis:		Es kann kein Linienmuster übergeben werden.

K.4.8 GetScanLine (\$c13c)

Berechnet das erste Byte einer Grafikzeile innerhalb des Grafikbildschirms.

Übergabe:	x	Grafikzeile (0-199).
Rückgabe:		dispBufferOn:
		Bit 7=1 und Bit 6=1: Adresse im Vordergrundbildschirm.
	r5	Adresse im Vordergrundbildschirm.
	r6	Adresse im Hintergrundbildschirm.
		Bit 7=0 und Bit 6=1: r5/r6 enthalten die Adresse im Hintergrundbildschirm.
		Bit 7=1 und Bit 6=0: r5/r6 enthalten die Adresse im Vordergrundbildschirm.
		GEOS V2: Bit 7=0 und Bit 6=0: r5/r6 enthalten die Mitte des Vordergrundbildschirm.
		Nur MP3: Bit 7=0 und Bit 6=0, r5/r6 enthalten die Adresse im Vordergrundbildschirm.
Verwendet:	dispBufferOn	Bit 7=1: Adresse Vordergrundbildschirm berechnen. Bit 6=1: Adresse Hintergrundbildschirm berechnen.
Verändert:	a	
Unverändert:	x	Grafikzeile (0-199).
Hinweis:		Unter GEOS128 ist die Grafikadresse im 80-Zeichen-Modus innerhalb des VDC-RAM.

K.4.9 SetPattern (\$c139)

Neues Füllmuster für gefüllte Rechtecke festlegen.

Übergabe: a Füllmuster (0-31).

Rückgabe: curPattern Zeiger auf Musterdaten.

Verändert: a

Hinweis: Um die Muster 32+33 anzusprechen setzt man curPattern auf das Muster 31 und addiert den Wert 8 für Muster 32 oder den Wert 16 für Muster 33.

K.4.10 FrameRectangle (\$c127)

Zeichnet ein unausgefülltes Rechteck mit einem 1-Pixel breiten Rand.

Übergabe: r2L Byte, y-Koordinate, obere Seite des Rechtecks.
r2H Byte, y-Koordinate, untere Seite des Rechtecks.
r3 Word, x-Koordinate, linke Seite des Rechtecks.
r4 Word, x-Koordinate, rechte Seite des Rechtecks.

a Byte, die 8 Bit definieren das Linienmuster.

Verwendet: dispBufferOn Bit 7=1: In Vordergrund schreiben.
Bit 6=1: In Hintergrund schreiben.
Kombination von Bit 7 und Bit 6 möglich.

Verändert: a, x, y
r5 bis r9, r11

Unverändert: r2L, r2H, r3, r4

K.4.11 i_FrameRectangle (\$c1a2)

Zeichnet ein unausgefülltes Rechteck mit einem 1-Pixel breiten Rand.

Inlinedaten: jsr
b yOben Byte, y-Koordinate, obere Seite des Rechtecks.
b yUnten Byte, y-Koordinate, untere Seite des Rechtecks.
w xLinks Word, x-Koordinate, linke Seite des Rechtecks.
w xRechts Word, x-Koordinate, rechte Seite des Rechtecks.
b Muster Byte, die 8 Bit definieren das Linienmuster.
... Das Programm wird ab hier fortgesetzt.

Rückgabe: r2L Byte, y-Koordinate, oberes Ende des Rechtecks.
r2H Byte, y-Koordinate, unteres Ende des Rechtecks.
r3 Word, x-Koordinate, linkes Ende des Rechtecks.
r4 Word, x-Koordinate, rechtes Ende des Rechtecks.

Verwendet: dispBufferOn Bit 7=1: In Vordergrund schreiben.
Bit 6=1: In Hintergrund schreiben.
Kombination von Bit 7 und Bit 6 möglich.

Verändert: a, x, y
r5 bis r9, r11
returnAddress Rücksprungadresse für Inline-Routine.

K.4.12 Rectangle (\$c124)

Zeichnet ein ausgefülltes Rechteck mit einem zuvor festgelegten Füllmuster.

Übergabe:	r2L r2H r3 r4	Byte, y-Koordinate, obere Seite des Rechtecks. Byte, y-Koordinate, untere Seite des Rechtecks. Word, x-Koordinate, linke Seite des Rechtecks. Word, x-Koordinate, rechte Seite des Rechtecks.
Verwendet:	dispBufferOn curPattern	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich. Über SetPattern definiertes Füllmuster.
Verändert:	a, x, y r5 bis r8 r11L r11H	r11L=Zeilenzähler. r11H=Füllmuster berechnen (HorizontalLine).
Unverändert:	r2L, r2H, r3, r4	
Aufruf von:	HorizontalLine	

K.4.13 i_Rectangle (\$c19f)

Zeichnet ein ausgefülltes Rechteck mit einem zuvor festgelegten Füllmuster.

Inlinedaten:	jsr b yOben b yUnten w xLinks w xRechts ...	Byte, y-Koordinate, obere Seite des Rechtecks. Byte, y-Koordinate, untere Seite des Rechtecks. Word, x-Koordinate, linke Seite des Rechtecks. Word, x-Koordinate, rechte Seite des Rechtecks. Das Programm wird ab hier fortgesetzt.
Rückgabe:	r2L r2H r3 r4	Byte, y-Koordinate, oberes Ende der Linie. Byte, y-Koordinate, unteres Ende der Linie. Word, x-Koordinate, linkes Ende der Linie. Word, x-Koordinate, rechtes Ende der Linie.
Verwendet:	dispBufferOn curPattern	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich. Über SetPattern definiertes Füllmuster.
Verändert:	a, x, y r5 bis r8 r11L r11H returnAddress	r11L=Zeilenzähler. r11H=Füllmuster berechnen (HorizontalLine). Rücksprungadresse für Inline-Routine.
Aufruf von:	HorizontalLine	

K.4.14 InvertRectangle (\$C12a)

Invertiert ein Rechteck im Vorder- und/oder Hintergrundbildschirm.

Übergabe:	r2L	Byte, y-Koordinate, obere Seite des Rechtecks.
	r2H	Byte, y-Koordinate, untere Seite des Rechtecks.
	r3	Word, x-Koordinate, linke Seite des Rechtecks.
	r4	Word, x-Koordinate, rechte Seite des Rechtecks.
Verwendet:	dispBufferOn	Bit 7=1: In Vordergrund schreiben.
		Bit 6=1: In Hintergrund schreiben.
		Kombination von Bit 7 und Bit 6 möglich.
		Bit 7=1 und Bit 6=1: Grafik im Vordergrund wird invertiert und in den Hintergrund kopiert.
Verändert:	a, x, y r5 bis r8, r11L	Bit 7=0 und Bit 6=0: Nur Vordergrundbildschirm. (siehe GetScanLine)
		r11L=Zeilenzähler.
Unverändert:	r2L, r2H, r3, r4	

K.4.15 ImprintRectangle (\$C250)

Kopiert ein Rechteck aus dem Vordergrund- in den Hintergrundbildschirm.

Übergabe:	r2L	Byte, y-Koordinate, obere Seite des Rechtecks.
	r2H	Byte, y-Koordinate, untere Seite des Rechtecks.
	r3	Word, x-Koordinate, linke Seite des Rechtecks.
	r4	Word, x-Koordinate, rechte Seite des Rechtecks.
Verändert:	a, x, y	
	r5 bis r8, r11L	r11L=Zeilenzähler.
Unverändert:	r2L, r2H, r3, r4	

K.4.16 i_ImprintRectangle (\$C253)

Kopiert ein Rechteck aus dem Vordergrund- in den Hintergrundbildschirm.

Inlinedaten:	jsr	
	b yOben	Byte, y-Koordinate, obere Seite des Rechtecks.
	b yUnten	Byte, y-Koordinate, untere Seite des Rechtecks.
	w xLinks	Word, x-Koordinate, linke Seite des Rechtecks.
	w xRechts	Word, x-Koordinate, rechte Seite des Rechtecks.
Rückgabe:	...	Das Programm wird ab hier fortgesetzt.
	r2L	Byte, y-Koordinate, obere Seite des Rechtecks.
	r2H	Byte, y-Koordinate, untere Seite des Rechtecks.
	r3	Word, x-Koordinate, linke Seite des Rechtecks.
	r4	Word, x-Koordinate, rechte Seite des Rechtecks.
Verändert:	a, x, y	
	r5 bis r8, r11L	r11L=Zeilenzähler.
	returnAddress	Rücksprungadresse für Inline-Routine.

K.4.17 RecoverRectangle (\$c12d)

Kopiert ein Rechteck aus dem Hintergrund- in den Vordergrundbildschirm.

Übergabe:	r2L	Byte, y-Koordinate, obere Seite des Rechtecks.
	r2H	Byte, y-Koordinate, untere Seite des Rechtecks.
	r3	Word, x-Koordinate, linke Seite des Rechtecks.
	r4	Word, x-Koordinate, rechte Seite des Rechtecks.
Verändert:	a, x, y	
	r5 bis r8, r11L	r11L=Zeilenzähler.
Unverändert:	r2L, r2H, r3, r4	

K.4.18 i_RecoverRectangle (\$c1a5)

Kopiert ein Rechteck aus dem Hintergrund- in den Vordergrundbildschirm.

Inlinedaten:	jsr	
	b yOben	Byte, y-Koordinate, obere Seite des Rechtecks.
	b yUnten	Byte, y-Koordinate, untere Seite des Rechtecks.
	w xLinks	Word, x-Koordinate, linke Seite des Rechtecks.
	w xRechts	Word, x-Koordinate, rechte Seite des Rechtecks.
	...	Das Programm wird ab hier fortgesetzt.
Rückgabe:	r2L	Byte, y-Koordinate, obere Seite des Rechtecks.
	r2H	Byte, y-Koordinate, untere Seite des Rechtecks.
	r3	Word, x-Koordinate, linke Seite des Rechtecks.
	r4	Word, x-Koordinate, rechte Seite des Rechtecks.
Verändert:	a, x, y	
	r5 bis r8, r11L	r11L=Zeilenzähler.
	returnAddress	Rücksprungadresse für Inline-Routine.

K.4.19 GraphicsString (\$c136)

Führt Grafikbefehle aus, die in einem String-ähnlichen Format abgelegt sind.

Übergabe:	r0	Word, Zeiger auf Befehlsstring.
Verwendet:	dispBufferOn	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich.
Verändert:	a, x, y r0 bis r13	

Befehlsübersicht:

::01h	MOVEPENTO w xPos b yPos	; Position Grafik-Cursor setzen ; Word, x-Koordinate in Pixel ; Byte, y-Koordinate in Pixel
::02h	LINETo w xPos b yPos	; Linie zeichnen ; Word, x-Koord. Endpunkt Linie ; Byte, y-Koord. Endpunkt Linie
::03h	RECTANGLETo w xPos b yPos	; Ausgefülltes Rechteck zeichnen ; Word, x-Koord. zweiter Eckpunkt ; Byte, y-Koord. zweiter Eckpunkt
::04h	PENFILL	; Ohne Funktion
::05h	NEWPATTERN b PATTERN	; Füllmuster für RECTANGLETo setzen ; Byte, Füllmuster 0-31 möglich!
::06h	ESC_PUTSTRING w xPos b yPos b "Text...",NULL	; Übergabe an i_PutString ; Word, x-Koordinate/Pixel ; Byte, y-Koordinate/Baseline/Pixel ; Text, Ende mit NULL-Byte
::07h	FRAME_RECTO w xPos b yPos	; Rahmen um Rechteck zeichnen ; Word, x-Koord. zweiter Eckpunkt ; Byte, y-Koord. zweiter Eckpunkt
::08h	PEN_X_DELTA w xDelta	; Grafik-Cursor verschieben ; Word, x-Position/Pixel
::09h	PEN_Y_DELTA b yDelta	; Grafik-Cursor verschieben ; Byte, y-Position/Pixel
::0Ah	PEN_X_DELTA w xDelta b yDelta	; Grafik-Cursor verschieben ; Word, x-Position/Pixel ; Byte, y-Position/Pixel
::00h	NULL	; Ende GraphicsString

K.4.20 i_GraphicsString (\$c1a8)

Führt Grafikbefehle aus, die in einem String-ähnlichen Format abgelegt sind.

Inlinedaten:	jsr b GRAPHICS b NULL ...	Grafikdaten. Ende mit NULL-Byte. Das Programm wird ab hier fortgesetzt.
Verwendet:	dispBufferOn	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich.
Verändert:	a, x, y r0 bis r13 returnAddress	Rücksprungadresse für Inline-Routine.

K.4.21 BitmapUp (\$c142)

Entpackt eine komprimierte Grafik und zeigt diese am Bildschirm an.

Übergabe:	r0 r1L r1H r2L r2H	Word, Zeiger auf Bitmap-Daten. Byte, x-Koordinate in Cards. Byte, y-Koordinate in Pixel. Byte, Breite der Grafik in Cards. Byte, Höhe der Grafik in Pixel.
Verwendet:	dispBufferOn	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich.
Verändert:	a, x, y r0 bis r9L	
Hinweis:		Die Bitmap darf max. Bildschirmgröße haben!

K.4.22 i_BitmapUp (\$c1ab)

Entpackt eine komprimierte Grafik und zeigt diese am Bildschirm an.

Inlinedaten:	jsr w BITMAP b xLinks b xOben b xBreite b xHöhe ...	Word, Zeiger auf Bitmap-Daten, max. Bildschirmgröße! Byte, x-Koordinate in Cards. Byte, y-Koordinate in Pixel. Byte, Breite der Grafik in Cards. Byte, Höhe der Grafik in Pixel. Das Programm wird ab hier fortgesetzt.
Verwendet:	dispBufferOn	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich.
Verändert:	a, x, y r0 bis r9L returnAddress	Rücksprungadresse für Inline-Routine.
Hinweis:		Die Bitmap darf max. Bildschirmgröße haben!

K.4.23 BitmapClip (\$c2aa)

Entpackt eine komprimierte Grafik und zeigt einen Ausschnitt am Bildschirm an.

Übergabe:	r0	Word, Zeiger auf Bitmap-Grafik.
	r1L	Byte, x-Koordinate in Cards.
	r1H	Byte, y-Koordinate in Pixel.
	r2L	Byte, Breite des Ausschnitts in Cards.
	r2H	Byte, Höhe des Ausschnitts in Pixel.
	r11L	Byte, Anzahl Cards vor dem Ausschnitt.
	r11H	Byte, Anzahl Cards nach dem Ausschnitt.
Verwendet:	r12	Word, Anzahl Pixelzeilen vor dem Ausschnitt.
	dispBufferOn	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich.
Verändert:	a, x, y r0 bis r12	
Hinweis:		$r11L + r2L + r11H$ = Breite der Bitmap-Grafik!

K.4.24 BitOtherClip (\$c2c5)

Entpackt eine komprimierte Grafik und zeigt einen Ausschnitt am Bildschirm an.

Übergabe:	r1L	Byte, x-Koordinate in Cards.
	r1H	Byte, y-Koordinate in Pixel.
	r2L	Byte, Breite des Ausschnitts in Cards.
	r2H	Byte, Höhe des Ausschnitts in Pixel.
	r11L	Byte, Anzahl Cards vor dem Ausschnitt.
	r11H	Byte, Anzahl Cards nach dem Ausschnitt.
	r12	Word, Anzahl Pixelzeilen vor dem Ausschnitt.
	r13	Word, Zeiger auf Eingaberoutine.
	r14	Word, Zeiger auf Sync-Routine.
Verwendet:	dispBufferOn	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich.
Verändert:	a, x, y r0 bis r12	
Unverändert:	r13, r14	
Hinweis:		$r11L + r2L + r11H$ = Breite der Bitmap-Grafik! Eingaberoutine muss Byte in die Adresse speichern, auf die das Register r0 zeigt und darf die Register r0 bis r14 selbst nicht verändern.

Sync-Routine:

```
:SYNC      LoadW  r0,Buf134      ; Zeiger auf Datenpuffer setzen
            rts
```


K.4.25 NormalizeX (\$C128)

Umrechnen einer x-Koordinate an den 40- oder 80-Zeichen-Bildschirm.

Übergabe: x Zeiger auf Zeropage-Register mit x-Koordinate
Rückgabe: Register Zeropage-Register enthält angepasste x-Koordinate
Verwendet: graphMode Bit 7=1 (80-Zeichen-Modus):

Positive Zahlen:

Wenn in x-Koordinate DOUBLE_W (Bit 15) gesetzt ist, dann Verdopplung ausführen.

Wenn Bit 14 gelöscht ist und ADD1_W (Bit 13) gesetzt, dann x-Koordinate um 1 Pixel erweitern.

Negative Zahlen:

Wenn in x-Koordinate DOUBLE_W (Bit 15) gelöscht ist, dann Verdopplung ausführen.

Wenn Bit 14 gesetzt ist, dann wird bei Verdopplung von negativen Zahlen der Wert ADD1_W abgezogen.

Bei negativen Zahlen muss das DOUBLE_W- und ADD1_W-Bit nicht über eine ODER-Verknüpfung gesetzt werden, sondern über EOR invertiert werden.

EOR wird vom MegaAssembler nicht unterstützt!

Bit 7=0 (80-Zeichen-Modus):

Bit 13 bis Bit 15 werden im Highbyte gelöscht.

Verändert: a

Unverändert: Register

Wenn Bit 15=0 und Bit 14=0 oder Bit 15=1 und Bit 14=1, dann wird der Wert nicht normalisiert.

Hinweis:

Auf Grund eines Fehlers in der Kernal-Routine von GEOS128 funktioniert das normalisieren nicht bei negativen Zahlen:

Die Beschreibung im "Hitchhikers Guide to GEOS" erklärt die Funktionsweise von Bit 13-15 und wie negative Zahlen behandelt werden sollen, aber im Kernalcode fehlen zwei notwendige Assemblerbefehle.

In GEOS/MegaPatch128 nach dem 21.12.2022 ist ein Patch enthalten, der auch negative Zahlen korrekt verdoppeln kann.

K.4.26 SetNewMode (\$c2dd; C128)

Bildschirm auf den 40- oder 80-Zeichen-Modus umschalten.

Übergabe:	graphMode	Bit 7=1: Auf 80-Zeichen-Modus umschalten. Bit 7=0: Auf 40-Zeichen-Modus umschalten.
Rückgabe:	graphMode	Neuer Bildschirm-Modus.
Verändert:	a, x, y r0	
	r0 bis r15 r0	Angabe "The Hitchhikers Guide to GEOS". Angabe "The Hitchhikers Guide to GEOS v2022".
	r2 bis r8	Nur MP3: Durch setzen der Bildschirmfarben.
	MP3-Farbtabelle	Nur MP3: Die Farbtabelle ab \$9fea bis \$9fff wird auf den neuen Bildschirm-Modus angepasst.
	rightMargin mouserRight	40Z=319, 80Z=639. 40Z=319, 80Z=639.
	CLKRATE (\$d030)	CPU-Taktfrequenz anpassen: 40Z=1MHz, 80Z=2MHz.
Aufruf von:	UseSystemFont	Zeichensatz für 40Z-/80Z-Modus aktivieren

K.4.27 SetColorMode (\$c2f5; C128)

Farb-Modus für den 80-Zeichen-Bildschirm setzen.

Alternativ:	VDC_ModelInit	
Übergabe:	a	Werte von 0-4 möglich: Nur MP3: 0/1 = Modus 2 setzen. 0 = 640 * 200 Pixel, keine Farbe (ATR aus). 1 = 640 * 176 Pixel, Farbe: 8*8 Pixel. 2 = 640 * 200 Pixel, Farbe: 8*8 Pixel. 3 = 640 * 200 Pixel, Farbe: 8*4 Pixel. 4 = 640 * 200 Pixel, Farbe: 8*2 Pixel.
Verändert:	a, x, y r0	
	r2 bis r8	Nur MP3: Durch setzen der Bildschirmfarben.
Verwendet von:	FirstInit StartAppl	
Hinweis:		Nur MP3: Auch als VDC_ModelInit bezeichnet.

K.4.28 ColorCard (\$C2f8; C128)

Farbe für ein Card im 40-/80-Zeichen-Bildschirm setzen oder auslesen.

Alternativ: ColorPoint

Übergabe: r3 Word, x-Koordinate des Card.
 r11L Nur GEOSV2: Byte, y-Koordinate des Card.
 r2L Nur MP3: Byte, y-Koordinate des Card.
 Carry-Flag C=0: Farbwert des Card auslesen.
 C=1: Farbwert des Card setzen.

Verwendet: graphMode Bit 7=1: Farbe im 80-Zeichen-Modus setzen.
 Bit 7=0: Farbe im 40-Zeichen-Modus setzen.

Verändert: a, x, y
 r5 Nur GEOSV2.

Unverändert: r3 Nur GEOSV2.
 r11L Nur MP3.
 r2L

Verwendet von: ColorRectangle Farb-Rechteck zeichnen.

Hinweis: Unter GEOSV2 wird die Verdopplung der x-Koordinate nicht unterstützt. Unter MP3 wird statt r11L das Register r2L für die y-Koordinate verwendet.

Die Routine sollte nicht mehr verwendet werden!

K.4.29 ColorRectangle (\$C2fb; C128)

Farb-Rechteck im 40-/80-Zeichen-Bildschirm zeichnen.

Übergabe: r3 Word, linke x-Koordinate in Pixel.
 r4 Word, rechte x-Koordinate in Pixel.
 r2L Byte, obere y-Koordinate in Pixel.
 r2H Byte, untere y-Koordinate in Pixel.

Verwendet: graphMode Bit 7=1: Farbe im 80-Zeichen-Modus setzen.
 Bit 7=0: Farbe im 40-Zeichen-Modus setzen.

Verändert: a, x, y
 r11L Nur GEOS128 V2.
 r5 bis r8 Nur MP3: Zur Umwandlung der Koordinaten in Cards.

Unverändert: r2 bis r4

Aufruf von: ColorCard Einzelnes Card einfärben.

Hinweis: Unter GEOS128 V2 wird die Verdopplung der x-Koordinate nicht unterstützt.

Die Routine arbeitet sehr langsam und sollte daher nicht verwendet werden!

K.4.30 RecColorBox (\$c0e5; MP3)

Farb-Rechteck im 40-/80-Zeichen-Bildschirm zeichnen.

Übergabe:	r5L	Byte, linke x-Koordinate in Cards.
	r5H	Byte, rechte x-Koordinate in Cards.
	r6L	Byte, obere y-Koordinate in Cards.
	r6H	Byte, untere y-Koordinate in Cards.
	r7L	Farbwert.
Verändert:	a, x, y	
	r5 bis r8	

K.4.31 i_ColorBox (\$c0df; MP3)

Farb-Rechteck im 40-/80-Zeichen-Bildschirm zeichnen.

Inlinedaten:	jsr	
	b xLinks	Byte, linke x-Koordinate in Cards.
	b yOben	Byte, rechte x-Koordinate in Cards.
	b xBreite	Byte, Breite in Cards.
	b yHöhe	Byte, Höhe in Cards.
	b FARBE	Farbwert.
	...	Das Programm wird ab hier fortgesetzt.
Verändert:	a, x, y	
	r5 bis r8	

K.4.32 i_UserColor (\$c0dc; MP3)

Farb-Rechteck im 40-/80-Zeichen-Bildschirm zeichnen.

Inlinedaten:	lda #FARBE	
	jsr	
	b xLinks	Byte, linke x-Koordinate in Cards.
	b yOben	Byte, rechte x-Koordinate in Cards.
	b xBreite	Byte, Breite in Cards.
	b yHöhe	Byte, Höhe in Cards.
	...	Das Programm wird ab hier fortgesetzt.
Verändert:	a, x, y	
	r5 bis r8	

K.4.33 DirectColor (\$c0e2; MP3)

Farb-Rechteck im 40-/80-Zeichen-Bildschirm zeichnen.

Übergabe:	r2L	Byte, obere y-Koordinate in Pixel.
	r2H	Byte, untere y-Koordinate in Pixel.
	r3	Word, linke x-Koordinate in Pixel.
	r4	Word, rechte x-Koordinate in Pixel.
	a	Farbwert.
Verändert:	a, x, y	
	r5 bis r8	
Aufruf von:	RecColorBox	

K.4.34 GetBackScreen (\$c0e8; MP3)

Hintergrundbild laden oder Bildschirm löschen.

Übergabe:

-

Verwendet:

sysRAMFlg

Bit 3=1: Hintergrundbild darstellen.

Bit 3=0: Bildschirm löschen.

BackScrPattern

Füllmuster für Bildschirm, wenn Bit 3=0.

Verändert:

a, x, y

r0 bis r8, r11

Aufruf von:

FetchRAM

Hintergrundbild aus REU laden.

SetPattern

i_Rectangle

Bildschirm löschen wenn Bit 3=0.

K.4.35 ResetScreen (\$c0eb; MP3)

Bildschirm initialisieren und mit Muster #2 und Hintergrundfarbe füllen.

Übergabe:

-

Verwendet:

graphMode

Nur MP3-128, Bildschirm-Modus abfragen.

C_GEOS_MOUSE

Farbe Mauszeiger.

C_GEOS_FRAME

Rahmenfarbe.

C_GEOS_BACK

Bildschirmfarben.

Verändert:

a, x, y

r0 bis r8, r11

mob0clr

Farbe Mauszeiger.

mob1clr

Farbe Textcursor.

extclr

Rahmenfarbe.

dispBufferOn

Nur in Vordergrund schreiben.

screencolors

Bildschirmfarben.

Aufruf von:

SetPattern

i_UserColor

i_Rectangle

K.5 Textroutinen

GEOS-Routinen für die Textausgabe.

K.5.1 PutChar (\$c145)

Ausgabe eines ASCII-Zeichen oder eines Steuercode auf dem Bildschirm.

Übergabe:	a	ASCII-Code.
	r11	Word, x-Koordinate in Pixel.
	r1H	Byte, y-Koordinate in Pixel.
Rückgabe:	r11	Word, x-Koordinate in Pixel für das nächste Zeichen.
	r1H	Byte, y-Koordinate in Pixel für das nächste Zeichen.
Verwendet:	dispBufferOn	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich.
	windowTop windowBottom leftMargin rightMargin	Fenstergrenzen für Textausgabe.
	StringFaultVec	Routine, wenn linker/rechter Rand überschritten wird.
	currentMode	Verändert durch LoadCharSet: Aktueller Schriftstil.
	baselineOffset	Position der Grundlinie im aktuellen Zeichensatz.
	curSetWidth	Länge einer Bitstream-Reihe in Bytes.
	curHeight	Anzahl Bitstream-Zeilen im Zeichensatz.
	curIndexTable	Zeiger auf aktuelle Index-Tabelle.
	cardDataPtr	Zeiger auf erste Bitstream-Reihe.
Verändert:	a, x, y	
	r1L, r2 bis r10 r12, r13	
Hinweis:		Die Ausgabe von Steuercodes zwischen \$00-\$07 bzw. \$1c-\$1f führt zu unvorhersehbaren Ergebnissen, bis hin zu einem "Systemfehler nahe \$..."

Steuercodes:

::08h	b BACKSPACE	; Vorheriges Zeichen löschen
::09h	b FORWARDSPACE	; Ohne Funktion
::0Ah	b LF	; Cursor um eine Zeile nach unten
::0Bh	b HOME	; Cursor nach links/oben, Achtung: ; Baseline=\$0000, 1xLF empfohlen
::0Ch	b UPLINE	; Cursor um eine Zeile nach oben
::0Dh	b CR	; Cursor auf Anfang nächste Zeile

::0Eh	b ULINEON	; Text unterstreichen
::0Fh	b ULINEOFF	; Unterstreichen abschalten
::12h	b REV_ON	; Text invertiert ausgeben
::13h	b REV_OFF	; Invertierung abschalten
::18h	b BOLDON	; Text in Fettschrift ausgeben
::19h	b ITALICON	; Text in Kursivschrift ausgeben
::1Ah	b OUTLINEON	; Text in Konturschrift ausgeben
::1Bh	b PLAINTEXT	; Alle Textformatierungen löschen

K.5.2 SmallPutChar (\$c202)

Ausgabe eines ASCII-Zeichen auf dem Bildschirm.

Übergabe:	a r11 r1H	ASCII-Code, nur Werte von \$20-\$80. Word, x-Koordinate in Pixel. Byte, y-Koordinate in Pixel.
Rückgabe:	r11	Word, x-Koordinate in Pixel für das nächste Zeichen. Clipping am linken Bildschirmrand mit negativer x-Koordinate.
Verwendet:	/*	siehe PutChar
Verändert:	a, x, y r1L, r2 bis r10 r12, r13	
Unverändert:	r1H	Byte, y-Koordinate in Pixel.
Hinweis:		Unter GEOS128 V2 wird die Verdopplung der x-Koordinate nicht unterstützt. Clipping/linker Rand erst ab GEOS V1.4.

K.5.3 PutDecimal (\$c184)

Ausgabe von vorzeichenlosen Integerzahlen.

Übergabe:	r0	Word, Integerzahl 0-65535.
	a	Formatierungsbyte: Bit 7=1: SET_LEFTJUST, linksbündig ausgeben. Bit 7=0: SET_RIGHTJUST, rechtsbündig ausgeben. Bit 6=1: SET_SUPRESS, nicht mit "0" auffüllen. Bit 6=0: SET_NOSUPRESS, mit "0" auf 5 Zeichen auffüllen. Bit 5=0: Größe Ausgabefeld in Pixel (0-63 / RIGHTJUST).
	r11	Word, x-Koordinate in Pixel.
	r1H	Byte, y-Koordinate in Pixel.
Rückgabe:	r11	Word, x-Koordinate in Pixel für das nächste Zeichen.
Verwendet:	/*	siehe PutChar
Verändert:	a, x, y r0, r1L, r2 bis r10 r12, r13	
Unverändert:	r1H	Byte, y-Koordinate in Pixel.

K.5.4 PutString (\$c148)

Ausgabe eines Strings aus ASCII-Zeichen auf dem Bildschirm.

Übergabe:	r0	Zeiger auf Textstring, Ende mit NULL-Byte.
	r11	Word, x-Koordinate in Pixel.
	r1H	Byte, y-Koordinate in Pixel.
Rückgabe:	r11	Word, x-Koordinate in Pixel für das nächste Zeichen.
	r1H	Byte, y-Koordinate in Pixel für das nächste Zeichen.
	r0	Word, Zeiger auf NULL-Byte des Strings.
Verwendet:	*/*	siehe PutChar
Verändert:	a, x, y	
	r0, r1L, r2 bis r10	
	r12, r13	

SteuerCodes:

::10h	b ESC_GRAPHICS	; Übergabe an GraphicsString
	b ...	; Daten für GraphicsString
::14h	b GOTOX	; Neue x-Koordinate setzen
	w xPos	; Word, neue x-Koordinate
::15h	b GOTOY	; Neue y-Koordinate setzen
	b yPos	; Byte, neue y-Koordinate
::16h	b GOTOXY	; Neue x- und y-Koordinate setzen
	w xPos	; Word, neue x-Koordinate
	b yPos	; Byte, neue y-Koordinate
::17h	b NEWCARDSET	; Drei Byte überspringen
	b \$00,\$00,\$00	; Dummy-Bytes
::80h	b SHORTCUT	; Nur BSW9/128: C= Zeichen ausgeben ; (Shortcut-Symbol in Menüs)

K.5.5 i_PutString (\$c1ae)

Ausgabe eines Inline-Strings auf dem Bildschirm.

Inlinedaten:	jsr	
	w xPos	Word, x-Koordinate in Pixel.
	b yPos	Byte, y-Koordinate in Pixel.
	b "Text..."	Textstring, auch SteuerCodes.
	b NULL	Endekennung.
	...	Das Programm wird ab hier fortgesetzt.
Rückgabe:	r11	Word, x-Koordinate in Pixel für das nächste Zeichen.
	r1H	Byte, y-Koordinate in Pixel für das nächste Zeichen.
	r0	Word, Zeiger auf NULL-Byte des Strings.
Verwendet:	*/*	siehe PutChar
Verändert:	a, x, y	
	r0, r1L, r2 bis r10	
	r12, r13	
	returnAddress	Rücksprungadresse für Inline-Routine.

K.5.6 GetString (\$c1ba)

Eingabe von Text über die Tastatur.

Übergabe:	r0	Zeiger auf Textstring als Vorgabe für die Texteingabe, Ende mit NULL-Byte. Nach [RETURN] auch Speicher für Texteingabe.
	r1L	Fehlerflag: Bit 7=0: Systemroutine verwenden.
	r4	Bit 7=1: Benutzerroutine in r4 verwenden. Fehlerroutine wird aufgerufen, wenn der Cursor den rechten Rand in rightMargin erreicht hat.
	r2L	Max. erlaubte Zeichenanzahl.
	r11	Word, x-Koordinate in Pixel für Texteingabe.
	r1H	Byte, obere y-Koordinate in Pixel für Texteingabe, nicht wie bei PutChar die Grundlinie der Zeichen!
Verwendet:	dispBufferOn	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich.
	windowTop windowBottom leftMargin rightMargin	Fenstergrenzen für Textausgabe.
	keyVector	Routine, die nach [RETURN] aufgerufen wird. Das Programm wird über diese Routine fortgesetzt.
	StringFaultVec	Routine, wenn linker/rechter Rand überschritten wird.
	currentMode	Aktueller Schriftstil.
	baselineOffset	Position der Grundlinie im aktuellen Zeichensatz.
	curSetWidth	Länge einer Bitstream-Reihe in Bytes.
	curHeight	Anzahl Bitstream-Zeilen im Zeichensatz.
	curIndexTable	Zeiger auf aktuelle Index-Tabelle.
	cardDataPtr	Zeiger auf erste Bitstream-Reihe.
Verändert:	a, x, y r0 bis r13	

K.5.7 GetNextChar (\$c2a7)

Liest das nächste Zeichen aus dem Tastaturpuffer ein.

Übergabe:	n/a	
Rückgabe:	a	Tastencode des Zeichens. \$00 wenn kein weiteres Zeichen im Tastaturpuffer.
Verwendet:	pressFlag	Bit 7=0: Kein Zeichen im Tastaturpuffer.
Verändert:	a, x	

K.5.8 UseSystemFont (\$c14b)

Aktiviert den Zeichensatz BSW9 (40Z-Modus) oder BSW128 (80Z-Modus).

Übergabe:	n/a	
Rückgabe:	baselineOffset curSetWidth curHeight curlIndexTable cardDataPtr	Position der Grundlinie im BSW-Zeichensatz. Länge einer Bitstream-Reihe in Bytes. Anzahl Bitstream-Zeilen im Zeichensatz. Zeiger auf aktuelle Index-Tabelle. Zeiger auf erste Bitstream-Reihe.
Aufruf von:	LoadCharSet	
Verändert:	a, y, r0	x-Register wird nicht verändert.

K.5.9 GetCharWidth (\$c1c9)

Ermittelt die Breite eines vorgegebenen Zeichens.

Übergabe:	a	ASCII-Code des Zeichen (\$20 bis \$7e).
Rückgabe:	a	Breite des ASCII-Zeichen. Zeichencode \$00 bis \$1f: Breite = \$00. Zeichencode \$7f: Breite wie letztes Zeichen, wird aus CurCharWidth (C64:\$8807, C128:\$880d) übernommen. Zeichencode \$20 bis \$7e und \$80: Breite wird aus der Index-Tabelle im Zeichensatz berechnet.
Verwendet:	curlIndexTable	Zeiger auf aktuelle Index-Tabelle.
Verändert:	y	

K.5.10 GetRealSize (\$c1b1)

Ermittelt die Breite, Höhe und Grundlinie eines vorgegebenen Zeichens.

Übergabe:	a	ASCII-Code des Zeichen, \$20-\$80.
	x	Schriftstil wie in currentMode: Bit 7=1: Unterstreichen (underline). Bit 6=1: Fettschrift (bold). Bit 5=1: Inversschrift (invert). Bit 4=1: Kursivschrift (italic). Bit 3=1: Konturschrift (outline). Bit 2=1: Hochgestellt (superscript). Bit 1=1: Tiefgestellt (subscript). Bit 0: Nicht verwendet.
Rückgabe:	a	Offset von oben bis zur Grundlinie, in Abhängigkeit vom vorgegebenen Schriftstil (bold +1, outline +2).
	x	Zeichensatzhöhe.
	y	Breite des aktuellen Zeichens.
Verwendet:	baselineOffset curlIndexTable	Offset von oben bis zur Grundlinie. Zeiger auf aktuelle Index-Tabelle.
Aufruf von:	GetCharWidth	Breite eines Zeichen ermitteln.

K.5.11 CmpString (\$c26b)

Vergleicht zwei, mit NULL-Byte abgeschlossene Strings.

Übergabe:	x	Lowbyte Zeropage-Adresse String 1.
	y	Lowbyte Zeropage-Adresse String 2.
Rückgabe:	Zero-Flag	Z=1: Beide Strings sind identisch. Z=0: Strings sind unterschiedlich.
	Carry-Flag	C=0: String 1 ist kleiner als String 2. C=1: String 1 ist größer als String 2.
	y	Bei Z=0: Zeiger auf erstes unterschiedliche Zeichen.
	Negative-Flag	Wenn ein Zeichen unterschiedlich ist: N=1: Zeichen in String 1 ist kleiner als String 2.
Verändert:	a, x, y	
Unverändert:	Zeropage	Adressen String 1 und String 2.
Hinweis:		Max. 256 Zeichen inkl. NULL-Byte!

K.5.12 CmpFString (\$c26e)

Vergleicht zwei Strings mit vorgegebener Länge.

Übergabe:	x	Lowbyte Zeropage-Adresse String 1.
	y	Lowbyte Zeropage-Adresse String 2.
	a	Stringlänge, max. 256 Zeichen.
Rückgabe:	Zero-Flag	Z=1: Beide Strings sind identisch. Z=0: Strings sind unterschiedlich.
	y	Bei Z=0: Zeiger auf erstes unterschiedliche Zeichen.
	Negative-Flag	Wenn ein Zeichen unterschiedlich ist: N=1: Zeichen in String 1 ist kleiner als String 2.
Verändert:	a, x, y	
Unverändert:	Zeropage	Adressen String 1 und String 2.

K.5.13 CopyString (\$c265)

Kopiert einen mit NULL-Byte abgeschlossene String.

Übergabe:	x	Lowbyte Zeropage-Adresse String 1.
	y	Lowbyte Zeropage-Adresse String 2.
Rückgabe:	y	Zeiger auf Byte hinter String 1.
Verändert:	a, x, y	
Unverändert:	Zeropage	Adressen String 1 und String 2.
Hinweis:		Bei überlappenden Bereichen muss der Ursprungstext weiter hinten im Speicher stehen, da sonst die hinteren Zeichen bereits ersetzt wurden bevor diese durch CopyString in den Zieltext kopiert wurden. Max. 256 Zeichen inkl. NULL-Byte!

K.5.14 CopyFString (\$c268)

Kopiert einen String mit vorgegebener Länge.

Übergabe:	x	Lowbyte Zeropage-Adresse String 1.
	y	Lowbyte Zeropage-Adresse String 2.
	a	Stringlänge, max. 256 Zeichen.
Rückgabe:	y	Zeiger auf Byte hinter String 1.
Verändert:	a, x, y	
Unverändert:	Zeropage	Adressen String 1 und String 2.
Hinweis:		Bei überlappenden Bereichen muss der Ursprungstext weiter hinten im Speicher stehen, da sonst die hinteren Zeichen bereits ersetzt wurden bevor diese durch CopyString in den Zieltext kopiert wurden.

K.5.15 PromptOn (\$c29b)

Textcursor einschalten.

Übergabe:	stringX	Word, x-Koordinate in Pixel für Textcursor.
	stringY	Byte, obere y-Koordinate in Pixel für Textcursor, nicht wie bei PutChar die Grundlinie der Zeichen!
Rückgabe:	n/a	
Verändert:	a, x	
	alphaFlag	Bit 6=1: Textcursor eingeschaltet.
	r3L	Sprite-Nr. für Textcursor.
	r4	x-Koordinate für Textcursor.
	r5L	y-Koordinate für Textcursor.
	r6	"Official GEOS Programmers Reference Guide", wird im GEOS-Kernal nicht verwendet.
Aufruf von:	PosSprite	Textcursor positionieren.
	EnablSprite	Textcursor einschalten.
Hinweis:		Vor dem Aufruf muss über InitTextPrompt der Textcursor initialisiert werden.

K.5.16 PromptOff (\$c29e)

Textcursor ausschalten.

Übergabe:	n/a	
Rückgabe:	n/a	
Verändert:	a, x, r3L	
Aufruf von:	DisablSprite	Textcursor ausschalten
Hinweis:		Interrupt über php,sei sperren, PromptOff aufrufen, alphaFlag auf \$00 setzen und Interrupt mittels plp zurücksetzen. Das verhindert das GEOS den Textcursor wieder einschaltet.

K.5.17 LoadCharSet (\$c1cc)

Neuen Zeichensatz für Zeichenausgabe festlegen.

Übergabe:	r0	Zeiger auf Zeichensatz.
Rückgabe:	baselineOffset	Position der Grundlinie im BSW-Zeichensatz.
	curSetWidth	Länge einer Bitstream-Reihe in Bytes.
	curHeight	Anzahl Bitstream-Zeilen im Zeichensatz.
	curIndexTable	Zeiger auf aktuelle Index-Tabelle.
	cardDataPtr	Zeiger auf erste Bitstream-Reihe.
Verändert:	a, y	
Unverändert:	r0	Zeiger auf Zeichensatz.

K.5.18 InitTextPrompt (\$c1c0)

Initialisiert den Textcursor.

Übergabe:	a	Höhe des Textcursor.
Rückgabe:	n/a	
Verwendet:	mob0clr	Farbe für Mauszeiger wird für Textcursor übernommen.
Verändert:	a, x, y	
	mob1clr	Farbe für Textcursor.
	moby2	Bit 1=0: Höhe max. 21 Pixel. Bit 1=1: Höhe max. 42 Pixel.
	alphaFlag	Bit 7=1: Nur alphanumerische Zeichen bei Texteingabe. Bit 0-5: \$03, Blinkfrequenz des Cursors.

K.6 Mousroutinen

GEOS-Routinen zur Steuerung des Mauszeigers.

K.6.1 StartMouseMode (\$c14e)

Mauszeiger initialisieren und Maussteuerung aktivieren.

Übergabe:	y r11 Carry-Flag	Byte, y-Koordinate für Mauszeiger. Word, x-Koordinate für Mauszeiger. C=1: Mausposition setzen, wenn r11<>\$0000. C=0: Mausposition nicht verändern.
Rückgabe:	n/a	
Verändert:	a, x, y, r0 bis r15 mouseXPos mouseYPos mouseVector mouseFaultVec faultData mouseOn mobenable	x-Koordinate für Mauszeiger. y-Koordinate für Mauszeiger. Routine zur Abfrage der Maustasten. Routine zur Bereichsprüfung. Wird auf \$00 gesetzt. Durch Aufruf von MouseUp. Sichtbarkeit eines Sprite, durch EnablSprite gesetzt.
Aufruf von:	SlowMouse MouseUp	Mauszeiger auf Anfangsgeschwindigkeit setzen. Mauszeiger einschalten.
Hinweis:		Der Mauszeiger wird erst über den Interrupt sichtbar!

K.6.2 MouseUp (\$c18a)

Mauszeiger sichtbar schalten.

Übergabe:	n/a	
Rückgabe:	n/a	
Verändert:	a mouseOn	Bit 7=1: Mauszeiger ist sichtbar.

K.6.3 MouseOff (\$c18d)

Mauszeiger unsichtbar schalten.

Übergabe:	n/a	
Rückgabe:	n/a	
Verändert:	a, x, y, r3L mouseOn mobenable	r3L=\$00: Sprite 0 = Mauszeiger abschalten. Bit 7=0: Mauszeiger ist unsichtbar. Sichtbarkeit eines Sprite, durch DisablSprite gesetzt.

K.6.4 ClearMouseMode (\$c19c)

Mauszeiger abschalten.

Übergabe:	n/a	
Rückgabe:	n/a	
Verändert:	a, x, y, r3L mouseOn mobenable	r3L=\$00: Sprite 0 = Mauszeiger abschalten. Bit 7=0: Mauszeiger ist unsichtbar. Sichtbarkeit eines Sprite, durch DisablSprite gesetzt.
Aufruf von:	DisablSprite	Sprite für Mauszeiger abschalten.

K.6.5 SlowMouse (\$fe83)

Setzt die Beschleunigung des Mauszeigers auf Anfang zurück.

Übergabe: n/a

Rückgabe: n/a

Verändert: a, x, y, r0 bis r15 Eingabetreiber dürfen alle Register verändern.

K.6.6 InitMouse (\$fe80)

Eingabetreiber nach dem Laden von Disk einmalig initialisieren.

Übergabe: n/a

Rückgabe: n/a

Verändert: a, x, y, r0 bis r15 Eingabetreiber dürfen alle Register verändern.
mouseXPos Word, x-Koordinate des Mauszeiger.
mouseYPos Byte, y-Koordinate des Mauszeiger.
inputData +0 Byte, aktuelle Bewegungsrichtung.
inputData +1 Byte, aktuelle Beschleunigung.
mouseData Tastenstatus.
pressFlag Änderung Tastenstatus und Bewegungsrichtung.

K.6.7 UpdateMouse (\$fe86)

Mausstatus abfragen.

Übergabe: n/a

Rückgabe: mouseXPos Eingabetreiber:
Word, x-Koordinate des Mauszeiger.
mouseYPos Byte, y-Koordinate des Mauszeiger.

inputData +0 Joysticktreiber (zusätzlich):
Byte, aktuelle Bewegungsrichtung.
inputData +1 Byte, aktuelle Beschleunigung.

mouseData Allgemein:
Bit 7=1: Keine Maustaste gedrückt.
Bit 7=0: Maustaste gedrückt.
pressFlag Bit 6=1: Bewegungsrichtung hat sich seit dem letzte Aufruf von UpdateMouse geändert.
Bit 5=1: Status der Maustasten hat sich seit dem letzten Aufruf von UpdateMouse geändert.

Verändert: a, x, y, r0 bis r15 Eingabetreiber dürfen alle Register verändern.

Verwendet von: GEOS-Interrupt Die GEOS-Interrupt-Routine fragt ca. 50-60x je Sekunde den Mausstatus über UpdateMouse ab.

Hinweis: Unter GEOS/MP3 setzt nur der Joystick-Treiber das Bit 6 von pressFlag, wenn sich die Richtung seit der letzten Abfrage geändert hat.
Ist das Bit gesetzt, dann wird durch die Mainloop die Routine aufgerufen, die in inputVector abgelegt ist.
Der Standardwert für inputVector ist \$0000, es wird also keine Routine aufgerufen.

K.6.8 SetMouse (\$fe89; C128)

Maustreiber nach Tastaturabfrage durch GEOS-Interrupt neu initialisieren.

Übergabe:	n/a	
Rückgabe:	n/a	
Verändert:	a, x, y, r0 bis r15	Eingabetreiber dürfen alle Register verändern.
	\$dc02	Alle Datenregister auf read/write schalten (Bit=1).
	\$dc00	Bit 6+7: %01 = Controlport 1.

K.6.9 SetMsePic (\$c2da; C128)

Mauszeiger-Bitmap definieren.

Übergabe:	r0	Zeiger auf Mauszeiger-Bitmap aus 2x16 Byte. Wenn r0=\$0000, dann Standard-Mauszeiger setzen.
		Aufbau der Mauszeiger-Bitmap: 16 Byte: AND-Maske für Mauszeiger. Eine 0 innerhalb des Byte löscht ein Bit in der Grafik. 16 Byte: OR-Maske für Mauszeiger. Eine 1 innerhalb des Byte setzt ein Bit in der Grafik.
Rückgabe:	n/a	
Verändert:	a, x, y r0 bis r4L r0 bis r15	"Ergänzungen zum MegaAssembler-Handbuch". Weitere Register werden im Kernal nicht verwendet. "The Hitchhikers Guide to GEOS".

K.6.10 TempHideMouse (\$c2d7; C128)

Entfernt vor Grafikaktionen alle Soft-Sprites im VDC-RAM vom Grafikbildschirm. Die Routine ist das Gegenstück zur Routine *DoSoftSprites*, die alle Soft-Sprites anzeigt.

Übergabe:	n/a	
Rückgabe:	n/a	
Verwendet:	graphMode	Bit 7=1: Mausgrafik entfernen. Bit 7=0: Routine beenden.
Verändert:	a, x, y	

K.6.11 HideOnlyMouse (\$c2f2; C128)

Nur Soft-Sprite für Mauszeiger vor Grafikaktionen im VDC-RAM entfernen.

Übergabe:	n/a	
Verwendet:	graphMode	Bit 7=1: Mausgrafik entfernen. Bit 7=0: Routine beenden.
Verändert:	a, x, y r1 bis r6	"GEOS-Programmierung mit dem MegaAssembler" und "The Hitchhikers Guide to GEOS". GEOS128 V2 und auch MP3-128 retten die Register r0 bis r6 auf den Stack und werden nicht verändert.

K.6.12 IsMselnRegion (\$c2b3)

Position des Mauszeigers überprüfen.

Übergabe:	r2L	Byte, y-Koordinate, obere Grenze des Bereichs.
	r2H	Byte, y-Koordinate, untere Grenze des Bereichs.
	r3	Word, x-Koordinate, linke Grenze des Bereichs.
	r4	Word, x-Koordinate, rechte Grenze des Bereichs.
Rückgabe:	a	TRUE (\$ff):
	Zero-Flag	Z=0: Mauszeiger innerhalb Bereich.
	a	FALSE (\$00):
	Zero-Flag	Z=1: Mauszeiger außerhalb Bereich.
Verändert:		Nur GEOS128:
	r3, r4	Durch NormalizeX werden die x-Koordinaten ggf. an den 40Z- oder 80Z-Modus angepasst.
Hinweis:		Der Interrupt sollte durch php-sei-plp während der Abfrage gesperrt werden, da sich sonst die Position des Mauszeiger während der Abfrage ändern könnte, was zu einem falschen Ergebnis führen kann.

K.7 Spriteroutinen

GEOS-Routinen zur Steuerung des Mauszeigers.

K.7.1 Allgemeines

Es können max. 8 Sprites dargestellt werden, inkl. Mauszeiger und ggf. Textcursor. Ein Sprite besteht aus einer Punktmatrix von 24x21 Pixel, das entspricht 63 Byte.

Unter GEOS128 werden 64 Byte erwartet, wobei in Byte 64 die tatsächlich genutzte Höhe in Pixelzeilen im Sprite definiert ist. Das letzte Byte wird als Information für den Soft-Sprite-Handler unter GEOS128, um nicht mehr Pixelzeilen darstellen zu müssen, als für das Sprite tatsächlich erforderlich sind.

Für die Behandlung des Mauszeigers und des Textcursor sollten die dafür vorgesehenen Routinen und nicht die folgenden Sprite-Routinen verwendet werden.

K.7.2 DrawSprite (\$c1c6)

Inhalt eines Sprite definieren.

Übergabe:	r3L	Byte, Sprite-Nummer (0-7).
	r4	Word, Zeiger auf Sprite-Definition: C64: 63 Byte C128: 63+1 Byte
Rückgabe:	n/a	
Verändert:	a, y, r5	
	spr0pic ... spr7pic	Ab spr0pic (\$8a00) liegen die 8x64 Byte für die Sprite-Definitionen der Sprites 0 bis 7.
Unverändert:	r3L, r4	
Hinweis:		Auch unter GEOS64 werden 64 Byte in die Sprite-Speicher ab spr0pic (\$8a00) kopiert, das letzte Byte wird von GEOS64 allerdings nicht verwendet.

K.7.3 PosSprite (\$c1cf)

Sprite am Bildschirm positionieren.

Übergabe:	r3L	Byte, Sprite-Nummer (0-7).
	r4	Word, x-Koordinate für Sprite-Position.
	r5L	Byte, y-Koordinate für Sprite-Position.
Rückgabe:	n/a	
Verändert:	a, x, y, r6	
Unverändert:	r3L, r4, r5L	

K.7.4 **EnablSprite (\$c1d2)**

Sprite am Bildschirm sichtbar machen.

Übergabe: r3L Byte, Sprite-Nummer (0-7).

Rückgabe: n/a

Verändert: a, x
 mobenble Sichtbarkeit eines Sprite.

Unverändert: r3L

K.7.5 **DisablSprite (\$c1d5)**

Sprite am Bildschirm unsichtbar machen.

Übergabe: r3L Byte, Sprite-Nummer (0-7).

Rückgabe: n/a

Verändert: a, x
 mobenble Sichtbarkeit eines Sprite.

Unverändert: r3L

K.7.6 **DoSoftSprites (\$e045; C128)**

Alle Soft-Sprites am Bildschirm darstellen. Die Routine ist das Gegenstück zur Routine *TempHideMouse*, die alle Soft-Sprites vom Bildschirm entfernt.

Übergabe: n/a

Rückgabe: n/a

Verändert: a, x, y
 r0 bis r5, r8 bis r12

K.8 Speicherverwaltungs-routinen

GEOS-Routinen zur Speicherverwaltung.

K.8.1 ClearRam (\$c178)

Speicherbereich mit NULL-Bytes löschen.

Übergabe:	r0	Word, Anzahl zu löschende Bytes.
	r1	Word, Zeiger auf Speicherbereich.
Rückgabe:	n/a	
Verändert:	a, y, r0, r1, r2L	
Hinweis:		Nicht auf den Bereich von r0 bis r2L anwenden.

K.8.2 FillRam (\$c17b)

Speicherbereich mit einem Byte-Wert füllen.

Übergabe:	r0	Word, Anzahl zu löschende Bytes.
	r1	Word, Zeiger auf Speicherbereich.
	r2L	Byte, Füllbyte.
Rückgabe:	n/a	
Verändert:	a, y, r0, r1	
Unverändert:	r2L	
Hinweis:		Nicht auf den Bereich von r0 bis r2L anwenden.

K.8.3 i_FillRam (\$c1b4)

Speicherbereich mit einem Byte-Wert füllen.

Inlinedaten:	jsr	
	w ANZAHL	Word, Anzahl zu löschende Bytes.
	w ADRESSE	Word, Zeiger auf Speicherbereich.
	b FILLBYTE	Byte, Füllbyte.
	...	Das Programm wird ab hier fortgesetzt.
Rückgabe:	r2L	Byte, Füllbyte.
Verändert:	a, y, r0, r1	
	returnAddress	Rücksprungadresse für Inline-Routine.
Hinweis:		Nicht auf den Bereich von r0 bis r2L anwenden.

K.8.4 InitRam (\$c181)

Ein oder mehrere Speicherbereiche initialisieren.

Übergabe:	r0	Word, Zeiger auf Initialisierungstabelle.
Rückgabe:	n/a	
Verändert:	a, x, y, r0, r1, r2L	
Hinweis:		Die Initialisierung darf nicht auf den Speicherbereich von r0 bis r2L angewendet werden.

Initialisierungstabelle für InitRam:

w Adresse	; Zeiger auf Speicherbereich
b Anzahl	; Anzahl Initialisierungsbytes
b \$xx, \$yy...	; Initialisierungsbytes
w Adresse	; Zeiger auf Speicherbereich
b Anzahl	; Anzahl Initialisierungsbytes
b \$xx, \$yy...	; Initialisierungsbytes
w NULL	; Abschluss-Word

K.8.5 MoveData (\$c17e)

Speicherbereich verschieben.

Übergabe: r0 Word, Zeiger auf Speicherbereich.
r1 Word, Zeiger auf Zielbereich.
r2 Word, Anzahl Bytes / Bereichsgröße.

Rückgabe: n/a

Verändert: a, x, y

Unverändert: r0 bis r2

Hinweis: Es können max. 64Kb verschoben werden.

Mit REU-Unterstützung unter GEOS64 können max. \$7900 Bytes verschoben werden, es findet keine Überprüfung der Bereichsgröße statt.

Unter GEOS128 können mit REU-Unterstützung max. \$3900 Bytes verschoben werden. Werden mehr Daten verschoben, dann wird ohne REU gearbeitet.

K.8.6 i_MoveData (\$c1b7)

Speicherbereich verschieben.

Inlinedaten: jsr
w ADRESSE Word, Zeiger auf Speicherbereich.
w ZIELADR Word, Zeiger auf Zielbereich.
w ANZAHL Word, Anzahl Bytes / Bereichsgröße.
... Das Programm wird ab hier fortgesetzt.

Rückgabe: r0 Word, Zeiger auf Speicherbereich.
r1 Word, Zeiger auf Zielbereich.
r2 Word, Anzahl Bytes / Bereichsgröße.

Verändert: a, x, y,
returnAddress Rücksprungadresse für Inline-Routine.

Hinweis: Bereichsgröße siehe MoveData.

K.8.7 MoveBData (\$c2e3; C128)

Speicherbereich zwischen FrontRAM und/oder BackRAM verschieben.

Übergabe:	r0	Word, Zeiger auf Speicherbereich 1.
	r3L	Byte, C128 Speicherbank 1.
	r1	Word, Zeiger auf Speicherbereich 2.
	r3H	Byte, C128 Speicherbank 2.
	r2	Word, Anzahl Bytes / Bereichsgröße.
Rückgabe:	n/a	
Verändert:	a, x, y	
Unverändert:	r0 bis r3	
Aufruf von:	DoBOP	%0000:0000: Verschieben von Speicher 1 nach 2.
Hinweis:		Speicherbank:
		\$00: BackRAM.
		\$01: FrontRAM (GEOS-Speicherbank).

K.8.8 DoBOP (\$c2ec; C128)

Speicherbereich im FrontRAM und/oder BackRAM bearbeiten.

Übergabe:	r0	Word, Zeiger auf Speicherbereich 1.
	r3L	Byte, C128 Speicherbank 1.
	r1	Word, Zeiger auf Speicherbereich 2.
	r3H	Byte, C128 Speicherbank 2.
	r2	Word, Anzahl Bytes / Bereichsgröße.
	y	Bearbeitungsmodus: %0000:0000: Verschieben von Speicher 1 nach 2. %0000:0001: Verschieben von Speicher 2 nach 1. %0000:0010: Vertauschen von Speicher 1 und 2. %0000:0011: Vergleichen von Speicher 1 und 2.
Rückgabe:	a / x	Ergebnis Vergleich:
	Zero-Flag = 0	\$ff: Speicher 1 und 2 unterschiedlich.
	Zero-Flag = 1	\$00: Speicher 1 und 2 identisch.
Verändert:	y	
	r0,r1 und r3L,r3H	Bearbeitungsmodus %0000:0001: r0/r1 und r3L/r3H werden vertauscht und anschließend wird Modus %0000:0000 ausgeführt.
Unverändert:	r2	Word, Anzahl Bytes / Bereichsgröße.
	r0, r1, r3L, r3H	Gilt nicht für Bearbeitungsmodus %0000:0001.
Aufruf durch:	MoveBData	%0000:0000: Verschieben von Speicher 1 nach 2.
	SwapBData	%0000:0010: Vertauschen von Speicher 1 und 2.
	VerifyBData	%0000:0011: Vergleichen von Speicher 1 und 2.
Hinweis:		Speicherbank:
		\$00: BackRAM.
		\$01: FrontRAM (GEOS-Speicherbank).
		Verschieben von Speicher 2 nach Speicher 1 funktioniert nicht mit GEOS128 und MP128 bis einschließlich 3.3r10.

K.8.9 SwapBData (\$c2e6; C128)

Speicherbereich im FrontRAM und BackRAM vertauschen.

Übergabe:	r0	Word, Zeiger auf Speicherbereich 1.
	r3L	Byte, C128 Speicherbank 1.
	r1	Word, Zeiger auf Speicherbereich 2.
	r3H	Byte, C128 Speicherbank 2.
	r2	Word, Anzahl Bytes / Bereichsgröße.
Rückgabe:	n/a	
Verändert:	a, x, y	
Unverändert:	r0 bis r3	
Aufruf von:	DoBOP	%0000:0010: Vertauschen von Speicher 1 und 2.
Hinweis:		Speicherbank:
		\$00: BackRAM.
		\$01: FrontRAM (GEOS-Speicherbank).

K.8.10 VerifyBData (\$c2e9; C128)

Speicherbereich im FrontRAM und BackRAM vergleichen.

Übergabe:	r0	Word, Zeiger auf Speicherbereich 1.
	r3L	Byte, C128 Speicherbank 1.
	r1	Word, Zeiger auf Speicherbereich 2.
	r3H	Byte, C128 Speicherbank 2.
	r2	Word, Anzahl Bytes / Bereichsgröße.
Rückgabe:	a / x	Ergebnis Vergleich:
	Zero-Flag = 0	\$ff: Speicher 1 und 2 unterschiedlich.
	Zero-Flag = 1	\$00: Speicher 1 und 2 identisch.
Verändert:	a, x, y	
Unverändert:	r0 bis r3	
Aufruf von:	DoBOP	%0000:0011: Vergleichen von Speicher 1 und 2.
Hinweis:		Speicherbank:
		\$00: BackRAM.
		\$01: FrontRAM (GEOS-Speicherbank).

K.8.11 DoRAMOp (\$c2d4)

Speicherbereich im RAM und/oder REU bearbeiten.

Übergabe:	r0	Word, Zeiger auf Speicher im RAM.
	r1	Word, Zeiger auf Speicher in REU.
	r2	Word, Anzahl Bytes / Bereichsgröße.
	r3L	Byte, Speicherbank in der REU.
	y	Bearbeitungsmodus: %10010000: Verschieben von RAM nach REU. %10010001: Verschieben von REU nach RAM. %10010010: Vertauschen von RAM und REU. %10010011: Vergleichen von RAM und REU.
Rückgabe:	x	\$00=Kein Fehler \$0d=DEV_NOT_FOUND, Speicherbank ungültig.
	a / y	Ergebnis Vergleich:
	Zero-Flag = 1 Zero-Flag = 0	Bit 5=0: Speicher identisch. Bit 5=1: Speicher unterschiedlich.
Verändert:	a, y	
Unverändert:	r0 bis r3L	
Hinweis:		Unter GEOS128 wird das FrontRAM verwendet.

K.8.12 StashRAM (\$c2c8)

Speicherbereich aus RAM nach REU verschieben.

Übergabe:	r0	Word, Zeiger auf Speicher im RAM.
	r1	Word, Zeiger auf Speicher in REU.
	r2	Word, Anzahl Bytes / Bereichsgröße.
	r3L	Byte, Speicherbank in der REU.
Rückgabe:	x	\$00=Kein Fehler. \$0d=DEV_NOT_FOUND, Speicherbank ungültig.
Verändert:	a, y	
Unverändert:	r0 bis r3L	
Hinweis:		Unter GEOS128 wird das FrontRAM verwendet.

K.8.13 FetchRAM (\$c2cb)

Speicherbereich aus REU nach RAM verschieben.

Übergabe:	r0	Word, Zeiger auf Speicher im RAM.
	r1	Word, Zeiger auf Speicher in REU.
	r2	Word, Anzahl Bytes / Bereichsgröße.
	r3L	Byte, Speicherbank in der REU.
Rückgabe:	x	\$00=Kein Fehler. \$0d=DEV_NOT_FOUND, Speicherbank ungültig.
Verändert:	a, y	
Unverändert:	r0 bis r3L	
Hinweis:		Unter GEOS128 wird das FrontRAM verwendet.

K.8.14 SwapRAM (\$c2ce)

Speicherbereich in RAM und REU vertauschen.

Übergabe:	r0	Word, Zeiger auf Speicher im RAM.
	r1	Word, Zeiger auf Speicher in REU.
	r2	Word, Anzahl Bytes / Bereichsgröße.
	r3L	Byte, Speicherbank in der REU.
Rückgabe:	x	\$00=Kein Fehler.
		\$0d=DEV_NOT_FOUND, Speicherbank ungültig.
Verändert:	a, y	
Unverändert:	r0 bis r3L	
Hinweis:	Unter GEOS128 wird das FronRAM verwendet.	

K.8.15 VerifyRAM (\$c2d1)

Speicherbereich in RAM und REU vergleichen.

Übergabe:	r0	Word, Zeiger auf Speicher im RAM.
	r1	Word, Zeiger auf Speicher in REU.
	r2	Word, Anzahl Bytes / Bereichsgröße.
	r3L	Byte, Speicherbank in der REU.
Rückgabe:	x	\$00=Kein Fehler.
		\$0d=DEV_NOT_FOUND, Speicherbank ungültig.
	a / y	Ergebnis Vergleich:
	Zero-Flag = 1	Bit 5=0: Speicher identisch.
	Zero-Flag = 0	Bit 5=1: Speicher unterschiedlich.
Verändert:	y	
Unverändert:	r0 bis r3L	
Hinweis:	Unter GEOS128 wird das FronRAM verwendet.	

K.9 GEOS-Rechenroutinen

GEOS-Routinen zum rechnen mit ganzen Zahlen.

K.9.1 Multiplikationsroutinen

K.9.1.1 BBMult (\$c160)

Zwei vorzeichenlose Bytes multiplizieren.

Übergabe:	x y	Zeiger auf Lowbyte ZeroPage-Register Faktor 1, Byte. Zeiger auf Lowbyte ZeroPage-Register Faktor 2, Byte.
Rückgabe:	Zeropage	Word, Ergebnis in Faktor 1 aus dem x-Register.
Verändert:	a, r7, r8 Faktor 1	
Unverändert:	x, y Faktor 2	

K.9.1.2 BMult (\$c163)

Vorzeichenloses Word mit vorzeichenlosen Byte multiplizieren.

Übergabe:	x y	Zeiger auf Lowbyte ZeroPage-Register Faktor 1, Word. Zeiger auf Lowbyte ZeroPage-Register Faktor 2, Byte.
Rückgabe:	Zeropage	Word, Ergebnis in Faktor 1 aus dem x-Register.
Verändert:	a, r6 bis r8 Faktor 1	
Unverändert:	x, y Faktor 2	

K.9.1.3 DMult (\$c166)

Zwei vorzeichenlose Word multiplizieren.

Übergabe:	x y	Zeiger auf Lowbyte ZeroPage-Register Faktor 1, Word. Zeiger auf Lowbyte ZeroPage-Register Faktor 2, Word.
Rückgabe:	Zeropage	Word, Ergebnis in Faktor 1 aus dem x-Register.
Verändert:	a, r6 bis r8 Faktor 1	
Unverändert:	x, y Faktor 2	

K.9.2 Divisionsroutinen

K.9.2.1 Ddiv (\$c169)

Zwei vorzeichenlose Word dividieren.

Übergabe:	x y	Zeiger auf Lowbyte ZeroPage-Register Faktor 1, Word. Zeiger auf Lowbyte ZeroPage-Register Faktor 2, Word.
Rückgabe:	Zeropage r8	Word, Ergebnis in Faktor 1 aus dem x-Register. Ganzzahliger Rest der Division.
Verändert:	a, r9 Faktor 1	
Unverändert:	x, y Faktor 2	
Hinweis:		Ddiv testet nicht ob der Divisor NULL ist.

K.9.2.2 DSdiv (\$c16c)

Zwei vorzeichenbehaftete Word dividieren.

Übergabe:	x y	Zeiger auf Lowbyte ZeroPage-Register Faktor 1, Word. Zeiger auf Lowbyte ZeroPage-Register Faktor 2, Word.
Rückgabe:	Zeropage r8	Word, Ergebnis in Faktor 1 aus dem x-Register. Ganzzahliger Rest der Division.
Verändert:	a, r9 Faktor 1	
Unverändert:	x, y Faktor 2	
Hinweis:		DSdiv testet nicht ob der Divisor NULL ist.

K.9.3 Rechenroutinen mit nur einem Operanden

K.9.3.1 Dabs (\$c16f)

Absoluten Wert eines vorzeichenbehafteten Word ermitteln.

Übergabe:	x	Zeiger auf Lowbyte ZeroPage-Register mit Word.
Rückgabe:	Zeropage	Word, Ergebnis in Adresse aus dem x-Register.
Verändert:	a	
Unverändert:	x	

K.9.3.2 Dnegate (\$c172)

Vorzeichen eines Word umdrehen.

Übergabe:	x	Zeiger auf Lowbyte ZeroPage-Register mit Word.
Rückgabe:	Zeropage	Word, Ergebnis in Adresse aus dem x-Register.
Verändert:	a	
Unverändert:	x	

K.9.3.3 Ddec (\$c175)

Inhalt eines Word um 1 dekrementieren.

Übergabe:	x	Zeiger auf Lowbyte ZeroPage-Register mit Word.
Rückgabe:	Zeropage Zero-Flag	Word, Ergebnis in Adresse aus dem x-Register. Z=1: Word hat den Wert NULL erreicht.
Verändert:	a	
Unverändert:	x	

K.9.3.4 DShiftLeft (\$c15d)

Inhalt eines Word um Anzahl Bit nach links schieben (Multiplikation um y^2).

Übergabe:	x y	Zeiger auf Lowbyte ZeroPage-Register mit Word. Anzahl der Linksverschiebungen.
Rückgabe:	Zeropage Carry-Flag	Word, Ergebnis in Adresse aus dem x-Register. Enthält das zuletzt aus dem Word geschobene Bit.
Verändert:	y	
Unverändert:	x	

K.9.3.5 DShiftRight (\$c262)

Inhalt eines Word um Anzahl Bit nach rechts schieben (Division um y^2).

Übergabe:	x y	Zeiger auf Lowbyte ZeroPage-Register mit Word. Anzahl der Rechtsverschiebungen.
Rückgabe:	Zeropage Carry-Flag	Word, Ergebnis in Adresse aus dem x-Register. Enthält das zuletzt aus dem Word geschobene Bit.
Verändert:	y	
Unverändert:	x	

K.10 Prozessroutinen

GEOS-Routinen zum verwalten von Hintergrundprozessen.

K.10.1 InitProcesses (\$c103)

Prozesstabelle initialisieren.

Übergabe: r0 Word, Zeiger auf Prozesstabelle.
a Anzahl Prozesse (1-20).

Rückgabe: n/a

Verändert: a, x, y, r1

ProcStatus X

Prozess-Status:
Bit 7=0: RUNABLE.
Bit 6=0: BLOCKED.
Bit 5=1: SET_FROZEN.
Bit 4=0: NOTIMER.

Unverändert: r0

Hinweis: Das NOTIMER_BIT wird vom Kernal nicht vollständig unterstützt. Es wird beim herunter zählen beachtet, aber nirgendwo gesetzt. In GDOS64 sysVersion=\$81 testweise integriert, für die Praxis nicht relevant.

Prozesstabelle:

```

w PROCROUT1      ; Adresse der Prozessroutine
w COUNT1         ; Anzahl Interrupts bis zur
                  ; nächsten Ausführung

w PROCROUT2      ; Adresse der Prozessroutine
w ...
```

K.10.2 RestartProcess (\$c106)

Einzelnen Prozess neu starten.

Übergabe: x Prozess-Nr. (0-19).

Rückgabe: n/a

Verändert: a

ProcStatus X

Prozess-Status:
Bit 6=0: BLOCKED.
Bit 5=0: FROZEN.

K.10.3 EnableProcess (\$c109)

Einzelnen Prozess neu starten.

Übergabe: x Prozess-Nr. (0-19).

Rückgabe: n/a

Verändert: a

ProcStatus X

Prozess-Status:
Bit 7=1: SET_RUNABLE.

K.10.4 BlockProcess (\$c10c)

Einzelnen Prozess von der Ausführung durch Mainloop sperren.

Übergabe: x Prozess-Nr. (0-19).
Rückgabe: n/a
Verändert: a
ProcStatus X Prozess-Status:
Bit 6=1: SET_BLOCKED.

K.10.5 UnblockProcess (\$c10f)

Einzelnen Prozess zur Ausführung durch die Mainloop freigeben.

Übergabe: x Prozess-Nr. (0-19).
Rückgabe: n/a
Verändert: a
ProcStatus X Prozess-Status:
Bit 6=0: BLOCKED.

K.10.6 FreezeProcess (\$c112)

Einzelnen Prozess anhalten und Zähler einfrieren.

Übergabe: x Prozess-Nr. (0-19).
Rückgabe: n/a
Verändert: a
ProcStatus X Prozess-Status:
Bit 5=1: SET_FROZEN.

K.10.7 UnfreezeProcess (\$c115)

Einzelnen Prozess freigeben und Zähler wieder runterzählen.

Übergabe: x Prozess-Nr. (0-19).
Rückgabe: n/a
Verändert: a
ProcStatus X Prozess-Status:
Bit 5=0: FROZEN.

K.10.8 Sleep (\$c199)

Programmroutine für einen bestimmten Zeitraum anhalten.

Übergabe: r0 Word, Wartezeit in 1/50s (PAL) bzw. 1/60s (NTSC).
Rückgabe: n/a
Verändert: a, x, y, r0 bis r15

Hinweis: Sleep muss in der obersten Ebene des Programms aufgerufen werden, da Sleep die beiden obersten Bytes vom Stack holt und zwischenspeichert.

Anschließend wird über den Befehl rts entweder zur übergeordneten Routine oder zur GEOS-MainLoop zurückgekehrt.

K.11 I/O-Routinen

GEOS-Routinen für den Zugriff auf Peripheriegeräte.

K.11.1 InitForIO (\$c25c)

Interrupt sperren, I/O-Bereich einblenden, CPU-Taktfrequenz auf 1MHz setzen.

Übergabe: n/a

Rückgabe: n/a

Verändert: a, y

Hinweis: InitForIO und DoneWithIO dürfen nicht geschachtelt werden, da wichtige Systemwerte gespeichert und ggf. überschrieben werden.

K.11.2 DoneWithIO (\$c25f)

I/O-Bereich ausblenden und CPU-Taktfrequenz und Interrupt-Status zurücksetzen.

Übergabe: n/a

Rückgabe: n/a

Verändert: a, y

Hinweis: InitForIO und DoneWithIO dürfen nicht geschachtelt werden, da wichtige Systemwerte gespeichert und ggf. überschrieben werden.

K.11.3 SetDevice (\$c2b0)

Neues Peripheriegerät (Laufwerk oder Drucker) aktivieren.

Übergabe: a Geräteadresse.

Rückgabe: x Fehlerstatus, \$00=Kein Fehler.

Verändert: a, y

curDrive Aktuelles GEOS-Laufwerk.
curDevice Aktuelle Geräteadresse.
curType Ab GEOS V1.3: Aktueller Laufwerkstyp aus driveType.

K.11.4 ChangeDiskDevice (\$c2bc)

Geräteadresse von zwei Laufwerken tauschen.

Übergabe: a Neue Geräteadresse.

Rückgabe: x Fehlerstatus, \$00=Kein Fehler.

Verwendet: curDrive Aktuelles GEOS-Laufwerk.

Verändert: a, y

r1 "Official GEOS Programmers Reference Guide".
Gilt nicht für alle Laufwerkstreiber, aber z.B. für den 1541-Treiber unter GEOS 64 V2 (r1L).

curDrive Neues GEOS-Laufwerk.
curDevice Neue Geräteadresse.

K.11.5 EnterTurbo (\$c214)

Aktiviert die Turbo-Software auf dem aktuellen Laufwerk.

Übergabe:	n/a	
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
Verwendet:	curDrive	Aktuelles GEOS-Laufwerk.
Verändert:	a, y	
	turboFlags	Bit 7=1: Turbo-Software installiert. Bit 6=1: Turbo-Software aktiv.

K.11.6 ExitTurbo (\$c232)

Deaktiviert die Turbo-Software auf dem aktuellen Laufwerk.

Übergabe:	n/a	
Rückgabe:	n/a	
Verwendet:	curDrive	Aktuelles GEOS-Laufwerk.
Verändert:	a, y	
	turboFlags	Bit 6=0: Turbo-Software inaktiv.

K.11.7 PurgeTurbo (\$c235)

Entfernt die Turbo-Software auf dem aktuellen Laufwerk.

Übergabe:	n/a	
Rückgabe:	n/a	
Verwendet:	curDrive	Aktuelles GEOS-Laufwerk.
Verändert:	a, x, y	
	r0 bis r3L	Nur bei schattiert arbeitenden Laufwerken durch Aufruf von StashRAM.
	r0 bis r3	"Official GEOS Programmers Reference Guide" und "The Hitchhikers Guide to GEOS".
	turboFlags	Bit 6=0: Turbo-Software inaktiv.

K.11.8 FirstInit (\$c271)

Initialisierung von GEOS.

Übergabe:	n/a	
Rückgabe:	n/a	
Verändert:	a, x, y	
	r0 bis r2L	Durch Verwendung von InitRam zur Initialisierung der GEOS-Systemwerte.
Hinweis:		Zusätzlich UseSystemFont aufrufen.

K.12 Diskettenroutinen

GEOS-Routinen für den Zugriff auf Laufwerke.

K.12.1 Die high-level-Diskettenroutinen

K.12.1.1 SetGEOSDisk (\$c1ea)

Die Diskette im aktuellen Laufwerk in eine GEOS-Diskette konvertieren.

Übergabe:	n/a	
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
	r0L, r1, r4 bis r5	
	r3	Durch SetNextFree für Suche nach einem freien Sektor für den Borderblock und durch CalcBlksFree für die Berechnung der Gesamtanzahl an Sektoren.
	curDirHead	Alle Laufwerke.
	dir2Head	1571, 1581, NativeMode.
	dir3Head	1581, NativeMode.
Aufruf von:	GetDirHead CalcBlksFree SetNextFree PutDirHead OpenRootDir OpenSubDir	Nur NativeMode, da die GEOS-Kennung immer im Hauptverzeichnis erzeugt wird.

K.12.1.2 GetPtrCurDkNm (\$c298)

Liefert einen Zeiger auf den Diskettenname für das aktuelle Laufwerk zurück.

Übergabe:	x	Zeiger auf Lowbyte ZeroPage-Register.
Rückgabe:	Zeropage	Word, Zeiger auf Diskettenname ohne NULL-Byte!
		Nur GEOS V1.2: Bei Laufwerk C: oder D: werden ungültige Werte übergeben, da für diese Laufwerke kein Puffer für den Diskettennamen existiert.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
	r7, r8, r15L	Nur GEOS V1.2: Werden durch den Aufruf von BBMult verändert.
Unverändert:	x	

K.12.1.3 OpenDisk (\$c2a1)

Diskette im aktuellen Laufwerk öffnen.

Übergabe:	n/a	
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
	r5	Zeiger auf Diskettenname.
	a	Nur MP3: CMD-RAMLink-Geräteadresse.
	y	Nur MP3: Aktive CMD-Partition.
Verwendet:	curDrive	Aktuelles Laufwerk.
	driveType	Für Laufwerke mit Shadow-Modus.
Verändert:	a, y, r1 bis r4	
	r5	Zeiger auf Diskettenname.
	r0	Nur bei Laufwerken mit Shadow-Modus.
	r6 bis r15	"Official GEOS Programmers Reference Guide".
	curDirHead	Alle Laufwerke.
	dir2Head	1571, 1581, NativeMode.
	dir3Head	1581, NativeMode.
	Dr?CurDkNm	Diskettenname.
	isGEOS	\$ff = GEOS-Diskette, \$00 = Keine GEOS-Diskette.
Aufruf von:	NewDisk	
	GetDirHead	
	GetPtrCurDkNm	
	ChkDkGEOS	

K.12.1.4 FindFTypes (\$c23b)

Liste mit bestimmten Dateitypen auf Diskette erstellen.

Übergabe:	r6	Word, Zeiger auf Ablagebereich für Dateinamen.
	r7H	Größe Ablagebereich = Anzahl Dateien x 17 Byte.
		Anzahl Dateien.
	r7L	Byte, GEOS-Filetyp.
		Nur MP3: GEOS-Filetyp=255: Alle Dateien einlesen.
	r10	Word, Zeiger auf GEOS-Klasse.
		Optional, \$0000 = GEOS-Klasse nicht testen.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
	r7H	Anzahl Dateien minus gefundene Dateien.
		Ist r7H unverändert, dann keine Datei gefunden.
	r5	Zeiger innerhalb diskBlkBuf auf Verzeichniseintrag der zuletzt gefundenen Datei durch GetNxtDirEntry.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
	r0 bis r2L, r4, r6	
	diskBlkBuf	Verzeichnisblock einlesen.
	fileHeader	Infoblock einlesen.
Aufruf von:	ClearRam	Ablagebereich für Dateinamen löschen.
	Get1stDirEntry	
	GetNxtDirEntry	

K.12.1.5 FindFile (\$c20b)

Datei im Verzeichnis auf Diskette suchen.

Übergabe:	r6	Word, Zeiger auf Dateiname.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
	dirEntryBuf	Wenn Datei gefunden: 30 Byte aus dem Verzeichniseintrag.
	r1L/r1H r5	Track/Sektor für Verzeichnisblock. Word, Zeiger auf Verzeichniseintrag in diskBlkBuf. (zeigt auf Byte #0 = CBM-Dateityp)
	diskBlkBuf	Verzeichnisblock.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y r4 bis r6 diskBlkBuf	Verzeichnisblock einlesen.
Aufruf von:	Get1stDirEntry GetNxtDirEntry	
Hinweis:		Nur MP3: Wird nach dem aktuellen Druckertreiber gesucht, dann erhält man im x-Register den Wert \$00, da der Druckertreiber immer im RAM vorgehalten wird. Die Register r1L/r1H und r5 sowie dirEntryBuf enthalten aber keine Angaben zum Verzeichniseintrag.

K.12.1.6 DeleteFile (\$c238)

Löscht eine Datei auf Diskette.

Übergabe:	r0	Word, Zeiger auf Dateiname.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y r1, r2, r4 bis r9 r0, r3	"Official GEOS Programmers Reference Guide" und "GEOS-Programmierung mit dem MegaAssembler".
	curDirHead dir2Head dir3Head	Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
	diskBlkBuf dirEntryBuf fileHeader	Verzeichnisblock einlesen. Datei über FindFile suchen. Indextabelle bei VLIR-Dateien.
Unverändert:	r0	Word, Zeiger auf Dateiname.
Aufruf von:	FindFile GetDirHead FreeBlock PutDirHead	

K.12.1.7 RenameFile (\$c259)

Ändert Dateiname einer Datei auf Diskette.

Übergabe:	r6	Word, Zeiger auf aktuellen Dateinamen.
	r0	Word, Zeiger auf neuen Dateinamen.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
		Wenn Datei gefunden wurde:
	r1L/r1H	Track/Sektor für Verzeichnisblock.
	r5	Word, Zeiger auf Verzeichniseintrag in diskBlkBuf. (zeigt auf Byte #0 = CBM-Dateityp)
	diskBlkBuf	Verzeichnisblock mit neuem Dateinamen.
Verwendet:	dirEntryBuf	Verzeichniseintrag mit ursprünglichem Dateinamen.
	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
	r1, r4 bis r6	
	diskBlkBuf	Verzeichnisblock einlesen.
	dirEntryBuf	Datei über FindFile suchen.
Unverändert:	r0	Word, Zeiger auf neuen Dateinamen.
Aufruf von:	FindFile	
Hinweis:		Es wird die erste, über FindFile gefundene Datei umbenannt. Es findet keine Prüfung statt ob der neue Dateiname bereits vergeben ist!

K.12.1.8 GetFile (\$c208)

Allgemeine Laderoutine um Dateien zu laden oder Programme zu starten.

Übergabe:	r6	Word, Zeiger auf Dateiname.
	r0L	Applications: Bit 7=1: Datenfile nachladen. Bit 6=1: Datenfile ausdrucken. Bit 0=1: Application nach r7 laden und nicht starten. Bit 0=0: Application an Ladeadresse laden und starten.
	r2	Wenn Bit 7=1 oder Bit 6=1: Word, Name der Diskette mit Datenfile. Wird nach dataDiskName kopiert und r2 dann auf dataDiskName gesetzt.
	r3	Word, Name des Datenfile. Wird nach dataFileName kopiert und r3 dann auf dataFileName gesetzt.
	r7	Wenn Bit 0=1: Word, Zeiger auf Ladeadresse.
	r10L	DeskAccessory: Byte, DAREcoverFlag, muss immer \$00 sein. Ursprünglicher Zweck: Bit 7=1: DA muss Vordergrund speichern/zurücksetzen. Bit 6=1: DA muss Farb-RAM speichern/zurücksetzen.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. Nur Application: Rückkehr zur Hauptanwendung nur wenn ein Diskfehler aufgetreten ist. Fehlerbehandlung siehe unten.
	r2 / r3 dirEntryBuf	Application: Zeiger auf Diskname / Datenfile, s.o. Zeiger auf Verzeichniseintrag.
Verwendet:	curDrive dlgBoxRamBuf	Aktuelles Laufwerk. Beim laden eines DeskAccessory.
Verändert:	a, y, r0 bis r10 diskBlkBuf fileHeader	
Aufruf von:	FindFile LdApplic LdDeskAcc LdFile	Application nachladen. DeskAccessory laden und starten. Datenfile laden.
Hinweis:		Nur MP3: Das SwapFile für DeskAccessories wird in der REU in der Datenbank (MP3_64K_DATA) abgelegt. Größe ist max. \$7c00 (MP64) oder \$6000 (MP128).

Fehlerbehandlung beim laden von Applications:

```

...                ; Parameter r0L, r2, r3 setzen
lda    #>EnterDesktop -1 ; Bei Ladefehler zurück zum
pha                ; Desktop wechseln
lda    #<EnterDesktop -1
pha
jmp    GetFile      ; Application laden und starten

```

K.12.1.9 SaveFile (\$c1ed)

Speicherbereiche auf Disk speichern oder leere VLIR-Dateien erzeugen.

Übergabe:	r9	Word, Zeiger auf Infoblock
		Byte 0+1 enthält einen Zeiger (Word) auf den Dateinamen der zu speichernden Datei, wird im Infoblock auf Diskette durch \$00,\$ff ersetzt.
		Byte 70 definiert Dateimodus (\$00=SEQ, \$01=VLIR).
		Sequentielle Datei: Byte 70+71 definiert die Ladeadresse und Byte 72+73 definiert die Endadresse +1 des Speicherbereichs der auf Diskette gespeichert werden soll. Bei einer Application definiert Byte 74+75 die Startadresse des Programms, die von der Ladeadresse abweichen kann, ansonsten wie Byte 70+71.
		VLIR-Datei Byte 70+71 und 74+75=\$0000, Byte 72+73=\$ffff.
Rückgabe:	r10L	Byte, erste Verzeichnisseite für Suche nach einem freien Verzeichniseintrag, erste Seite = \$00.
	x	Fehlerstatus, \$00=Kein Fehler.
	r6	Word, Zeiger auf fileTrScTab mit der Liste der reservierten Dateiblöcke.
Verwendet:	dirEntryBuf	Verzeichniseintrag der gespeicherten Datei.
	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
	r0	"Official GEOS Programmers Reference Guide" und "GEOS-Programmierung mit dem MegaAssembler". Wird vom GEOS-Kernal nicht verwendet.
	r1 bis r8	
	curDirHead	Alle Laufwerke.
	dir2Head	1571, 1581, NativeMode.
	dir3Head	1581, NativeMode.
	diskBlkBuf	Verzeichnisblock einlesen.
	fileTrScTab	Liste mit reservierten Dateiblöcken.
	fileHeader	Angepasster Infoblock zum speichern auf Diskette Byte 160 wird durch SaveFile gelöscht.
Unverändert:	r9	Word, Zeiger auf Infoblock.
Aufruf von:	GetDirHead	BAM einlesen.
	BlkAlloc	Dateiblöcke reservieren.
	SetGDirEntry	Verzeichniseintrag erstellen.
	PutDirHead	BAM aktualisieren.
	PutBlock	Infoblock speichern.

K.12.1.10 RstrAppl (\$c23e)

DeskAccessory beenden und zur Hauptanwendung zurückkehren.

Übergabe:	n/a	
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
		Nur GEOS 64 V2 und älter: Nur Fehler beim laden/löschen des SwapFile von Diskette werden zurückgemeldet.
		Nur GEOS 128 und MP3: Immer \$00=Kein Fehler (kein Zugriff auf Diskette).
Verwendet:	dlgBoxRamBuf	Diverse Speicherinhalte nach Ende des DeskAccessory wieder zurücksetzen.
	curDrive	Nur GEOS 64 V2 und älter: Laufwerk mit SwapFile.
Verändert:	a, x, y	
	r0 bis r10	Nur GEOS 64 V2 und älter: Durch GetFile zum laden des SwapFile.
	r0 bis r3L	Nur GEOS 128 / MP3: Durch MoveBData / FetchRAM.
Aufruf von:	GetFile	Nur GEOS 64 V2 und älter: SwapFile von Disk laden.
	FastDelFile	SwapFile auf Diskette löschen.
	MoveBData	Nur GEOS 128, SwapFile aus BackRAM laden.
	FetchRAM	Nur MP3: SwapFile aus REU laden.
Hinweis:		Ein DeskAccessory muss Laufwerkswechsel wieder rückgängig machen, damit unter GEOS64 V2 das SwapFile wieder eingelesen werden kann.
		Zusätzlich müssen ggf. die oberen 16 Pixelzeilen des Vordergrundbildschirms wieder hergestellt werden.
		Außerdem dürfen a0 bis a9 nicht verändert werden.

Empfohlene Routine zum beenden eines DeskAccessory:

:EndDA	Loadw	appMain, RstrAppl
	rts	

K.12.1.11 OpenRootDir (\$9050; MP3)

Öffnet auf einem NativeMode-Laufwerk das Hauptverzeichnis.

Übergabe:	n/a	-
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$40=Ungültiger Befehl (ILLEGAL DEVICE). Weitere Werte siehe OpenDisk.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y, r0 bis r5	Weitere Register siehe OpenDisk.
Aufruf von:	OpenDisk	
Hinweis:		"The Hitchhikers Guide to GEOS": Die Routine wird hier als OpenRoot bezeichnet.

K.12.1.12 OpenSubDir (\$9053; MP3)

Öffnet auf einem NativeMode-Laufwerk ein Unterverzeichnis.

Übergabe:	r1L/r1H	Track/Sektor des Verzeichnis-Headers.
	r1L=\$00	Setzt den Verzeichnisheader auf das Hauptverzeichnis ohne dabei OpenDisk auszuführen (erst ab MP3r11/14.3.2024).
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$40=Ungültiger Befehl (ILLEGAL DEVICE). Weitere Werte siehe OpenDisk.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y, r0 bis r5	Weitere Register siehe OpenDisk.
Aufruf von:	OpenDisk	
Hinweis:		"The Hitchhikers Guide to GEOS": Die Routine wird hier als OpenDirectory bezeichnet.

K.12.1.13 OpenPartition (\$9062; MP3)

Partition auf einem CMD-Laufwerk wechseln.

Übergabe:	r3H	Byte, Partitions-Nr.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$40=Ungültiger Befehl (ILLEGAL DEVICE). Weitere Werte siehe OpenDisk.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y, r0 bis r5	Weitere Register siehe OpenDisk.
Unverändert:	r3H	
Aufruf von:	ExitTurbo InitForIO SwapPartition DoneWithIO OpenDisk	
Hinweis:		Bei älteren Versionen von GEOS/MegaPatch muss zuvor auf NativeMode-Laufwerken OpenRootDir aufgerufen werden, da ansonsten bei einem geöffneten Unterverzeichnis das System in einer Endlosschleife festhängt. Es wird empfohlen, zuvor OpenRootDir oder OpenSubDir mit r1L=\$00 aufzurufen.

K.12.1.14 GetPDirEntry (\$905c; MP3)

Angaben aus Partitionsverzeichnis auf CMD-Laufwerk einlesen.

Übergabe:	r3H r4	Byte, Partitions-Nr. Word, Zeiger auf Speicher für Partitionsdaten (30 Byte).
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$40=Ungültiger Befehl (ILLEGAL DEVICE).
	r4	Word, Zeiger auf Speicher für Partitionsdaten (30 Byte).
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
Unverändert:	r3H, r4	
Aufruf von:	ExitTurbo InitForIO ReadPDirEntry DoneWithIO	

K.12.1.15 GetPTypeData (\$9068; MP3)

Erstellt Liste mit dem Typ (GEOS-Format) aller Partitionen auf CMD-Laufwerken.

Übergabe:	r4	Word, Zeiger auf Speicher für Partitionstypen (256 Byte).
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$40=Ungültiger Befehl (ILLEGAL DEVICE).
	r4	Word, Zeiger auf Speicher für Partitionstypen (256 Byte).
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
Aufruf von:	ExitTurbo InitForIO ReadPDirEntry DoneWithIO	

K.12.2 Die mid-level-Diskettenroutinen

K.12.2.1 NewDisk (\$c1e1)

Diskette im aktuellen Laufwerk initialisieren.

Übergabe:	n/a	
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
Verwendet:	curDrive curType	Aktuelles Laufwerk. Für Laufwerke mit Shadow-Modus.
Verändert:	a, y r1 r0, r2, r3	Nur bei Laufwerken mit Shadow-Modus zum initialisieren des Cache über DoRAMOp. r3H wird als Bankzähler verwendet.
Aufruf von:	EnterTurbo InitForIO DoneWithIO	Nur bei phys.Laufwerken um den INITIALIZE-Befehl an das TurboDOS im Laufwerk zu senden.

K.12.2.2 ChkDkGEOS (\$c1de)

Die Diskette im Laufwerk wird auf den GEOS-Status überprüft.

Übergabe:	r5	Zeiger auf die aktuelle BAM (curDirHead).
Rückgabe:	isGEOS a	\$ff = GEOS-Diskette, \$00 = Keine GEOS-Diskette. Gleicher Inhalt wie isGEOS.
Verwendet:	curDirHead	
Verändert:	x, y	
Unverändert:	r5	Zeiger auf die aktuelle BAM (curDirHead).
Verwendet von:	OpenDisk GetOPDPtr SetGEOSDisk	

K.12.2.3 CalcBlksFree (\$c1db)

Ermittelt die Anzahl freier Blöcke auf der aktuellen Diskette.

Übergabe:	r5	Zeiger auf die aktuelle BAM (curDirHead).
Rückgabe:	r4 r3	Word, Anzahl freier Blöcke auf Diskette. Ab GEOS V1.3: Word, Gesamtanzahl Blöcke auf Diskette.
Verwendet:	curDirHead dir2Head dir3Head	Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Verändert:	a, x, y	
Unverändert:	r5	Zeiger auf die aktuelle BAM (curDirHead).
Verwendet von:	OpenDisk	
Aufruf von:	GetBlock/ReadBlock	Nur NativeMode: BAM-Sektor einlesen. Erfordert TurboDOS
Hinweis:		Die BAM muss sich bereits im Speicher befinden.

K.12.2.4 GetDirHead (\$c247)

BAM von Diskette einlesen.

Übergabe:	n/a	
Rückgabe:	x r4	Fehlerstatus, \$00=Kein Fehler. Word, Zeiger auf curDirHead (1541 und NativeMode), dir2Head (1571) oder dir3Head (1581).
Verwendet:	curDrive. curDirHead dir2Head dir3Head	Aktuelles Laufwerk, Zeiger auf ersten BAM-Block. Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Verändert:	a, y, r1	
Verwendet von:	OpenDisk	
Aufruf von	GetBlock	
Hinweis:		1541/71: Wenn Diskette gewechselt werden kann, dann muss zuvor NewDisk aufgerufen werden (\$29,DSK_ID_MISMAT).

K.12.2.5 PutDirHead (\$c24a)

BAM im Speicher auf Diskette schreiben.

Übergabe:	n/a	
Rückgabe:	x r4	Fehlerstatus, \$00=Kein Fehler. Word, Zeiger auf curDirHead (1541 und NativeMode), dir2Head (1571) oder dir3Head (1581)
Verwendet:	curDrive, curDirHead dir2Head dir3Head	Aktuelles Laufwerk, Zeiger auf ersten BAM-Block. 1571, 1581, NativeMode. 1581, NativeMode.
Verändert:	a, y, r1	
Aufruf von:	PutBlock	

K.12.2.6 FindBAMBit (\$c2ad)

Status eines Diskettenblock in der BAM abfragen.

Übergabe:	r6L/r6H	Track/Sektor des gesuchten Block.
Rückgabe:	Zero-Flag x r8H r7H	Z=1: Block belegt (Bit für Sektor in BAM = 0). Zeiger auf Byte in curDirHead (1541), dir2Head (1571 oder NativeMode) oder dir3Head (1581) mit dem gesuchten Block. Bit-Maske zum isolieren des Block innerhalb des Byte mit dem AND-Befehl aus der BAM. Nur 1541: Zeiger auf Byte mit der Anzahl freier Blocks auf dem gesuchten Track.
Verwendet:	curDirHead dir2Head dir3Head	Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Verändert:	a, y	
Unverändert:	r6L/r6H	Track/Sektor des gesuchten Block.
Aufruf von:	GetBlock/ReadBlock	Nur NativeMode: BAM-Sektor einlesen. Erfordert TurboDOS
Hinweis:		Die BAM muss sich bereits im Speicher befinden.

K.12.2.7 BlkAlloc (\$c1fc)

Erstellt eine Liste mit reservierten Blocks für eine vorgegebene Anzahl an Bytes.

Übergabe:	r2	Word, Anzahl Bytes.
	r6	Word, Zeiger auf Ablagebereich für Block-Tabelle.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$03=Nicht genügend Speicher frei (INSUFF_SPACE).
Verwendet:	r2	Word, Anzahl reservierter Blocks.
	r3L/r3H	Track/Sektor des zuletzt reservierten Block.
	interleave	Sektorabstand, optimiert für GEOS-TurboDOS.
	curDirHead	Alle Laufwerke.
	dir2Head	1571, 1581, NativeMode.
Verändert:	dir3Head	1581, NativeMode.
	a, y, r4 bis r8	
Aufruf von:	NxtBlkAlloc	
	GetBlock/ReadBlock	Nur NativeMode: BAM-Sektor einlesen. Erfordert TurboDOS
Hinweis:		BAM ab ":curDirHead wird nicht automatisch gespeichert.
		Die Suche nach freien Sektoren beginnt immer ab dem ersten Datensektor auf der Diskette.

K.12.2.8 NxtBlkAlloc (\$c24d)

Erstellt eine Liste mit reservierten Blocks für eine vorgegebene Anzahl an Bytes.

Übergabe:	r2	Word, Anzahl Bytes.
	r3L/r3H	Es wird ab Track/Sektor nach freien Blöcken gesucht.
	r6	Word, Zeiger auf Ablagebereich für Block-Tabelle.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$03=Nicht genügend Speicher frei (INSUFF_SPACE).
Verwendet:	r2	Word, Anzahl reservierter Blocks.
	r3L/r3H	Track/Sektor des zuletzt reservierten Block.
	curDirHead	Alle Laufwerke.
	dir2Head	1571, 1581, NativeMode.
	dir3Head	1581, NativeMode.
Verändert:	a, y, r4 bis r8	
Aufruf von:	CalcBlksFree	
	SetNextFree	
	GetBlock/ReadBlock	Nur NativeMode: BAM-Sektor einlesen. Erfordert TurboDOS
Hinweis:		BAM ab ":curDirHead wird nicht automatisch gespeichert.

K.12.2.9 AllocateBlock (\$9048)

Einen einzelnen Block in der BAM als belegt markieren.

Übergabe:	r6L/r6H	Track/Sektor-Adresse des betreffenden Blocks.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$06=Block ist bereits belegt (BAD_BAM).
Verwendet:	curDirHead dir2Head dir3Head	Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Verändert:	a, y, r7 und r8H r7L	"GEOS-Programmierung mit dem MegaAssembler" und "Hitchhikers Guide to GEOS". Wird von GEOS / FindBAMBit nicht verändert.
Unverändert:	r6L/r6H	Track/Sektor-Adresse des betreffenden Blocks.
Aufruf von:	FindBAMBit GetBlock/ReadBlock	Nur NativeMode: BAM-Sektor einlesen. Erfordert TurboDOS
Hinweis:		Erst ab GEOS V1.5 mit KONFIGURIEREN V1.6! BAM ab "":curDirHead wird nicht automatisch gespeichert.

K.12.2.10 SetNextFree (\$c292)

Einen einzelnen Block in der BAM als belegt markieren.

Übergabe:	r3L/r3H	Es wird ab Track/Sektor nach freien Blöcken gesucht.
Rückgabe:	x r3L/r3H	Fehlerstatus, \$00=Kein Fehler. \$03=Nicht genügend Speicher frei (INSUFF_SPACE). Track/Sektor des zuletzt reservierten Block.
Verwendet:	interleave curDirHead dir2Head dir3Head	Sektorabstand, optimiert für GEOS-TurboDOS. Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Verändert:	a, y, r6, r7 und r8H	
Unverändert:	r6L/r6H	Track/Sektor-Adresse des betreffenden Blocks.
Aufruf von:	GetBlock/PutBlock	Nur NativeMode: BAM-Sektor einlesen. Erfordert TurboDOS
Hinweis:		BAM ab "":curDirHead wird nicht automatisch gespeichert.

K.12.2.11 FreeBlock (\$c2b9)

Einen einzelnen Block in der BAM als frei markieren.

Übergabe:	r6L/r6H	Track/Sektor-Adresse des betreffenden Blocks.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$06=Block ist bereits frei (BAD_BAM).
Verwendet:	curDirHead dir2Head dir3Head	Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Verändert:	a, y, r7H und r8H	
Unverändert:	r6L/r6H	Track/Sektor-Adresse des betreffenden Blocks.
Aufruf von:	GetBlock/ReadBlock	Nur NativeMode: BAM-Sektor einlesen. Erfordert TurboDOS
Hinweis:		Erst ab GEOS V1.3 im Kernall enthalten! BAM ab "":curDirHead wird nicht automatisch gespeichert.

K.12.2.12 FreeFile (\$c226)

Markiert alle Blöcke einer Datei in der BAM als frei.

Übergabe:	r9	Word, Zeiger auf Verzeichniseintrag. (z.B. dirEntryBuf)
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
Verändert:	a, y r1, r2, r4 bis r9 r0, r3	"Official GEOS Programmers Reference Guide".
	curDirHead dir2Head dir3Head diskBlkBuf fileHeader	Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Aufruf von:	GetDirHead PutDirHead	

K.12.2.13 GetFreeDirBlk (\$c1f6)

Freien Eintrag im Verzeichnis suchen und ggf. neuen Verzeichnisblock anlegen.

Übergabe:	r10L	Byte, Directory-Seite ab der gesucht werden soll.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$04=Verzeichnis voll (FULL DIRECTORY).
	r10L diskBlkBuf r1L/r1H r4 y-Register	Aktuelle Directory-Seite. Verzeichnisblock. Track/Sektor Verzeichnisblock. Zeiger auf diskBlkBuf. Zeiger auf 30-Byte-Bereich für Verzeichniseintrag innerhalb des Verzeichnisblock.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y r0, r3, r5, r7, r8 curDirHead dir2Head dir3Head	Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Verwendet von:	SetGDirEntry	
Hinweis:		BAM ab ":curDirHead wird nicht automatisch gespeichert.

K.12.2.14 SetGDirEntry (\$c1f0)

Verzeichniseintrag erzeugen und auf Diskette speichern.

Übergabe:	r10L	Byte, Directory-Seite ab der gesucht werden soll.
	r2	Word, Anzahl Blocks.
	r6	Word, Zeiger auf eine Block-Tabelle in fileTrScTab.
	r9	Word, Zeiger auf Infoblock (siehe SaveFile).
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$04=Verzeichnis voll (FULL DIRECTORY).
	r6	Word, Zeiger auf ersten nicht benötigten Block in der zuvor übergebenen Blocktabelle in fileTrScTab.
	dirEntryBuf	30-Byte Verzeichniseintrag.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	"Official GEOS Programmers Reference Guide" und "GEOS-Programmierung mit dem MegaAssembler".
	r1, r3 bis r5, r7, r8 r0, r2	
Aufruf von:	diskBlkBuf	
	BldGDirEntry	
	GetFreeDirBlk	
	PutDiskBlkBuf	
Hinweis:		Unter GEOS 64/128 muss r6 auf fileTrScTab zeigen.
		Nur MP3: Die Block-Tabelle in r6 kann auch in einem anderen Speicherbereich als fileTrScTab liegen.

K.12.2.15 BldGDirEntry (\$c1f3)

Verzeichniseintrag im Speicher erzeugen.

Übergabe:	r2	Word, Anzahl Blocks.
	r6	Word, Zeiger auf eine Block-Tabelle in fileTrScTab.
	r9	Word, Zeiger auf Infoblock (siehe SaveFile).
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$04=Verzeichnis voll (FULL DIRECTORY).
	r6	Word, Zeiger auf ersten nicht benötigten Block in der zuvor übergebenen Blocktabelle in fileTrScTab.
Verändert:	dirEntryBuf	30-Byte Verzeichniseintrag.
	a, y, r1H r3	Zeiger auf Dateiname (gleich Byte 0/1 im Infoblock).
Unverändert:	r9	Word, Zeiger auf Infoblock.
Hinweis:		Unter GEOS 64/128 muss r6 auf fileTrScTab zeigen.
		Nur MP3: Die Block-Tabelle in r6 kann auch in einem anderen Speicherbereich als fileTrScTab liegen.

K.12.2.16 Get1stDirEntry (\$9030)

Ersten Eintrag im Verzeichnis der Diskette einlesen.

Übergabe:	n/a	
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
	r1L/r1H	Track/Sektor des ersten Verzeichnisblocks.
	r4	Word, Zeiger auf diskBlkBuf.
	r5	Word, Zeiger auf den ersten Verzeichniseintrag innerhalb des Verzeichnisblocks.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
Aufruf von:	GetDiskBlkBuf	
Hinweis:		Erst ab GEOS V1.3 im Kernal enthalten!

K.12.2.17 GetNxtDirEntry (\$9033)

Nächsten Eintrag im Verzeichnis der Diskette einlesen.

Übergabe:	r1L/r1H	Track/Sektor des aktuellen Verzeichnisblocks.
	r5	Word, Zeiger auf aktuellen Eintrag.
	diskBlkBuf	Verzeichnisblock.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
	y	Ist der Wert <>0 wurde das Verzeichniseinde erreicht.
	r1L/r1H	Track/Sektor des aktuellen Verzeichnisblocks.
	r5	Word, Zeiger auf den nächsten Verzeichniseintrag innerhalb des Verzeichnisblocks.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, r4	Weitere Register wenn GetOPDPTr aufgerufen wird.
Aufruf von:	GetDiskBlkBuf	Verzeichnisblock einlesen.
	GetOPDPTr	Borderblock einlesen.
Hinweis:		Erst ab GEOS V1.3 im Kernal enthalten!

K.12.2.18 GetFHdrInfo (\$c229)

Infoblock einer Datei einlesen.

Übergabe:	r9	Word, Zeiger auf Verzeichniseintrag. (z.B. dirEntryBuf)
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
	fileHeader	Der Infoblock der Datei.
	r1L/r1H	Track/Sektor des ersten Blocks der sequentiellen Datei oder des VLIR-Indexblock.
	r7	Word, Zeiger auf Ladeadresse (gem. Infoblock).
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y, r4	
	fileTrScTab	Byte 0+1 enthalten Track/Sektor des Infoblock.
Unverändert:	r9	
Aufruf von:	GetBlock	

K.12.2.19 FollowChain (\$c205)

Erzeugt eine Tabelle mit allen Blöcken einer sequentiellen Datei.

Übergabe:	r1L/r1H r3	Track/Sektor des ersten Blocks der sequentiellen Datei. Word, Zeiger auf Speicherbereich für Track/Sektor-Tabelle (z.B. fileTrScTab, nur bis max. 127 Blocks).
Rückgabe:	x r1L/r1H	Fehlerstatus, \$00=Kein Fehler. Track/Sektor des letzten Blocks der Datei.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y, r4	
Unverändert:	r3	Word, Zeiger auf Speicher für Track/Sektor-Tabelle.
Aufruf von:	GetDiskBlkBuf	
Hinweis:		Die Routine liest auch mehr als 127 Blocks ein, der Speicherbereich, auf den r3 zeigt, muss daher groß genug sein um alle Blöcke aufnehmen zu können!

K.12.2.20 FastDelFile (\$c244)

Löscht eine Datei unter Verwendung einer Track/Sektor-Tabelle.

Übergabe:	r0 r3	Word, Zeiger auf Dateiname. Word, Zeiger auf Block-Tabelle.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y, r1, r4 bis r8 r2, r9	 "Official GEOS Programmers Reference Guide" und "GEOS-Programmierung mit dem MegaAssembler".
	curDirHead dir2Head dir3Head	Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Aufruf von:	GetDirHead PutDirHead	
Unverändert:	r0, r3	Word, Zeiger auf Dateiname / Block-Tabelle.

K.12.2.21 LdApplic (\$c21d)

Laderoutine um Applications zu starten.

Übergabe:	r9	Word, Zeiger auf Verzeichniseintrag. (z.B. dirEntryBuf)
	r0L	Bit 7=1: Datenfile nachladen Bit 6=1: Datenfile ausdrucken Bit 0=1: Application nach r7 laden und nicht starten Bit 0=0: Application an Ladeadresse laden und starten
	r2	Wenn r0L / Bit 7=1 oder Bit 6=1: Word, Name der Diskette mit Datenfile. Wird nach dataDiskName kopiert und r2 dann auf dataDiskName gesetzt.
	r3	Word, Name des Datenfile. Wird nach dataFileName kopiert und r3 dann auf dataFileName gesetzt.
	r7	Wenn r0L / Bit 0=1: Word, Zeiger auf Ladeadresse
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler
	r2 / r3	Application: Zeiger auf Diskname / Datenfile, s.o.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y r0 bis r15	
Aufruf von:	LdFile StartAppl	Datenfile laden Anwendung starten
Verwendet von:	GetFile	Allgemeine Laderoutine für GEOS.
Hinweis:		Rückkehr zur Hauptanwendung nur wenn ein Diskfehler aufgetreten ist. Da der Speicher zum Teil überschrieben sein könnte sollte hier die Rücksprungsadresse EnterDesktop-1 verwendet werden (siehe unten).

Fehlerbehandlung beim laden von Applications:

```

...                               ; Parameter r0L, r2, r3 setzen

lda    #>EnterDesktop -1 ; Bei Ladefehler zurück zum
pha                               ; Desktop wechseln
lda    #<EnterDesktop -1
pha

jmp    LdApplic                ; Application laden und starten

```

K.12.2.22 StartAppl (\$c22f)

GEOS initialisieren und anschließend eine Anwendung im Speicher starten.

Übergabe:	r7	Startadresse der Application im Speicher.
	r0L	Bit 7=1: Datenfile nachladen Bit 6=1: Datenfile ausdrucken Die Bit 0-5 werden nicht verwendet. Wenn Bit 7=1 oder Bit 6=1:
	r2	Word, Name der Diskette mit Datenfile Wird nach dataDiskName kopiert und r2 dann auf dataDiskName gesetzt
	r3	Word, Name des Datenfile Wird nach dataFileName kopiert und r3 dann auf dataFileName gesetzt
Rückgabe:	n/a	Keine Rückkehr zur Anwendung.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	n/a	
Aufruf von:	MainLoop	Anwendung ausführen.
Verwendet von:	GetFile	Allgemeine Laderoutine für GEOS.

K.12.2.23 LdDeskAcc (\$c217)

Laderoutine um DeskAccessories zu starten.

Übergabe:	r9	Word, Zeiger auf Verzeichniseintrag. (z.B. dirEntryBuf)
	r10L	Byte, DAREcoverFlag, muss immer \$00 sein. Ursprünglicher Zweck: Bit 7=1: Das DeskAccessory muss die Vordergrundgrafik speichern/zurücksetzen. Bit 6=1: Das DeskAccessory muss das Farb-RAM speichern/zurücksetzen.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
Verwendet:	dlgBoxRamBuf	
Verändert:	a, y r0 bis r15 fileHeader	
Aufruf von:	GetFHdrInfo ReadFile MainLoop	Infoblock einlesen für Größe des SwapFile. DeskAccessory laden. DeskAccessory ausführen.
Verwendet von:	GetFile	Allgemeine Laderoutine für GEOS.
Hinweis:		Nur MP3: Das SwapFile wird in der REU in der Datenbank (MP3_64K_DATA) abgelegt. Größe ist max. \$7c00 (MP64) oder \$6000 (MP128).

K.12.2.24 LdFile (\$c211)

Datenfile von Diskette in Speicher einlesen.

Übergabe:	r9	Word, Zeiger auf Verzeichniseintrag (z.B. dirEntryBuf). Interne GEOS-Register: Werden von GetFile oder LdApplic gesetzt und befinden sich im System-Bereich von GEOS.
	loadOpt loadAddr	
Rückgabe:	x r7	Fehlerstatus, \$00=Kein Fehler. Word, Zeiger auf zuletzt gelesenes Datenbyte +1.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y, r1, r4 fileHeader diskBlkBuf	
Unverändert:	r9	Word, Zeiger auf Verzeichniseintrag.
Aufruf von:	GetFHdrInfo GetDiskBlkBuf ReadFile	Infoblock einlesen für Dateistruktur SEQ oder VLIR. VLIR-Indexblock einlesen. Datenfile oder ersten VLIR-Datensatz einlesen.
Verwendet von:	GetFile	Allgemeine Laderoutine für GEOS.
Hinweis:		Nicht für den Einsatz in Programmen gedacht, da loadOpt / loadAddr nicht als Konstanten definiert sind (in GEOS64 und GEOS128 unterschiedlich) und die Parameter nicht direkt übergeben werden können.

K.12.2.25 ReadFile (\$c1ff)

Datenfile von Diskette in den Speicher einlesen.

Übergabe:	r1L/r1H r7 r2	Track/Sektor des ersten Blocks der sequentiellen Datei. Word, Zeiger auf Datenpuffer. Word, Größe des Datenpuffer in Bytes.
Rückgabe:	x fileTrScTab r2 r7 r1L/r1H	Fehlerstatus, \$00=Kein Fehler. Block-Tabelle mit Track/Sektor-Adressen. Word, Nicht genutzte Bytes im Datenpuffer. Word, Zeiger hinter das zuletzt gelesene Byte. Bei Fehler \$0b=BFR_OVERFLOW: Track/Sektor des letzten Blocks der nicht mehr in den Speicher eingelesen werden konnte.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y, r4 diskBlkBuf	
Aufruf von:	EnterTurbo InitForIO ReadBlock DoneWithIO	
Verwendet von:	ToBasic LdFile LdDeskAcc ReadRecord	
Hinweis:		ReadFile kann max. 32.258 Bytes einlesen, da fileTrScTab max. 127 Blocks aufnehmen kann.

K.12.2.26 WriteFile (\$c1f9)

Datenfile aus dem Speicher auf Diskette schreiben.

Übergabe:	r7 r6	Word, Zeiger auf Datenpuffer. Word, Zeiger auf Tabelle mit Track-/Sektoradressen.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y, r1, r4, r6, r7 diskBlkBuf	
	r2	Nach Angabe im "GEOS Reference Guide", wird im GEOS-Kernal aber nicht verwendet.
Aufruf von:	EnterTurbo InitForIO WriteBlock VerWriteBlock DoneWithIO	

Hinweis: WriteFile kann max. 32.258 Bytes speichern, da nur 127 Track-/Sektoradressen eingelesen werden können.

K.12.2.27 ReadByte (\$c2b6)

Einzelnes Byte aus einem Datenfile einlesen.

Übergabe:	r1L/r1H r4 r5L r5H	Track/Sektor des ersten Blocks der sequentiellen Datei. Word, Zeiger auf Datenpuffer. Byte, muss beim ersten Aufruf \$00 enthalten. Byte, muss beim ersten Aufruf \$00 enthalten.
Rückgabe:	x a r1L/r1H r5L r5H	Fehlerstatus, \$00=Kein Fehler. \$0b=Kein Byte mehr verfügbar (BFR_OVERFLOW). Gelesenes Byte aus dem Datenfile. Track/Sektor des aktuellen Blocks der Datei. Byte, Zeiger auf Byte im aktuellen Block. Byte, Anzahl Bytes im aktuellen Block.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	y	
Unverändert:	r4	Word, Zeiger auf Datenpuffer.
Aufruf von:	GetBlock	
Hinweis:		r1, r4 und r5 dürfen zwischen den Aufrufen von ReadByte nicht verändert werden.

K.12.2.28 ReadPDirEntry (\$905f; MP3)

Angaben aus Partitionsverzeichnis auf CMD-Laufwerk einlesen.

Übergabe:	r3H r4	Byte, Partitions-Nr. Word, Zeiger auf Speicherbereich für Partitionsdaten. (mind. 30 Bytes)
Rückgabe:	x r4	Fehlerstatus, \$00=Kein Fehler. \$40=Ungültiger Befehl (ILLEGAL DEVICE). Word, Zeiger auf Speicherbereich für Partitionsdaten.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
Unverändert:	r3H, r4	
Erfordert:	ExitTurbo InitForIO DoneWithIO	

Hinweis: Byte 0 = Partitionstyp im GEOS-Format!

K.12.2.29 SwapPartition (\$9065; MP3)

Partition auf einem CMD-Laufwerk wechseln.

Übergabe:	r3H	Byte, Partitions-Nr.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$40=Ungültiger Befehl (ILLEGAL DEVICE).
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y, r0 bis r5	
Unverändert:	r3H	Byte, Partitions-Nr.
Erfordert:	ExitTurbo InitForIO DoneWithIO	

Hinweis: Die Routine verändert nicht den Zeiger auf das aktuelle Unterverzeichnis. Es wird daher empfohlen auf NativeMode-Laufwerken zuvor OpenRootDir oder OpenSubDir mit r1L=\$00 aufzurufen um das Hauptverzeichnis zu aktivieren. Es wird anschließend kein OpenDisk ausgeführt!

K.12.3 Die low-level- und die very-low-level-Diskettenroutinen

K.12.3.1 ChangeDiskDevice (\$c2bc)

Geräteadresse von zwei Laufwerken tauschen.

Übergabe:	a	Neue Geräteadresse.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
Verwendet:	curDrive	Aktuelles GEOS-Laufwerk.
Verändert:	a, y r1	"Official GEOS Programmers Reference Guide". Gilt nicht für alle Laufwerkstreiber, aber z.B. für den 1541-Treiber unter GEOS 64 V2 (r1L).
	curDrive	Neues GEOS-Laufwerk.
	curDevice	Neue Geräteadresse.

K.12.3.2 ReadLink (\$904b)

Track-/Sektoradresse des nächsten Datenblock einlesen.

Übergabe:	r1L/r1H r4	Track/Sektor aktueller Block der sequentiellen Datei. Word, Zeiger auf Datenpuffer.
Rückgabe:	x r4	Fehlerstatus, \$00=Kein Fehler. Word, Zeiger auf Datenpuffer. In Byte 0+1 findet sich die Track-/Sektoradresse des nächsten Blocks.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	y	
Unverändert:	r1L/r1H r4	Track/Sektor aktueller Block der sequentiellen Datei. Word, Zeiger auf Datenpuffer.
Erfordert:	EnterTurbo InitForIO DoneWithIO	
Hinweis:		Erst ab GEOS V1.5 im GEOS-Kernal enthalten! Je nach Laufwerkstreiber werden entweder nur die beiden Link-Bytes oder der ganze Sektor eingelesen.

K.12.3.3 GetBlock (\$c1e4)

Einzelnen Block von Diskette einlesen.

Übergabe:	r1L/r1H r4	Track/Sektor-Adresse für Block. Word, Zeiger auf Datenpuffer.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
Unverändert:	r1L/r1H r4	Track/Sektor-Adresse für Block. Word, Zeiger auf Datenpuffer.
Aufruf von:	EnterTurbo InitForIO DoneWithIO	

K.12.3.4 ReadBlock (\$c21a)

Einzelnen Block von Diskette einlesen.

Übergabe:	r1L/r1H r4	Track/Sektor-Adresse für Block. Word, Zeiger auf Datenpuffer.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$02=Ungültige Block-Adresse (INV_BLOCK).
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
Unverändert:	r1L/r1H r4	Track/Sektor-Adresse für Block. Word, Zeiger auf Datenpuffer.
Erfordert:	EnterTurbo InitForIO DoneWithIO	

K.12.3.5 PutBlock (\$c1e7)

Einzelnen Block auf Diskette schreiben.

Übergabe:	r1L/r1H r4	Track/Sektor-Adresse für Block. Word, Zeiger auf Datenpuffer.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$02=Ungültige Block-Adresse (INV_BLOCK).
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
Unverändert:	r1L/r1H r4	Track/Sektor-Adresse für Block. Word, Zeiger auf Datenpuffer.
Aufruf von:	EnterTurbo InitForIO DoneWithIO	
Hinweis:	Nur 1571: Nach WriteBlock wird automatisch VerWriteBlock zur Überprüfung der Daten ausgeführt.	

K.12.3.6 WriteBlock (\$c220)

Einzelnen Block auf Diskette schreiben.

Übergabe:	r1L/r1H r4	Track/Sektor-Adresse für Block. Word, Zeiger auf Datenpuffer.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$02=Ungültige Block-Adresse (INV_BLOCK).
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
Unverändert:	r1L/r1H r4	Track/Sektor-Adresse für Block. Word, Zeiger auf Datenpuffer.
Erfordert:	EnterTurbo InitForIO DoneWithIO	

K.12.3.7 VerWriteBlock (\$c223)

Block auf Diskette vergleichen und ggf. erneut auf Diskette schreiben.

Übergabe:	r1L/r1H r4	Track/Sektor-Adresse für Block. Word, Zeiger auf Datenpuffer.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$02=Ungültige Block-Adresse (INV_BLOCK).
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
Unverändert:	r1L/r1H r4	Track/Sektor-Adresse für Block. Word, Zeiger auf Datenpuffer.
Erfordert:	EnterTurbo InitForIO DoneWithIO	
Aufruf von:	WriteBlock	
Hinweis:	Es empfiehlt sich zuerst alle Block auf Diskette zu schreiben und anschließend alle Blocks über VerWriteBlock zu überprüfen. Nur im 1541/1571-Laufwerkstreiber enthalten, alle anderen Treiber melden \$00=Kein Fehler zurück.	

K.12.3.8 GetDiskBlkBlock (\$903c)

Einzelnen Block von Diskette nach *diskBlkBuf* einlesen.

Übergabe:	r1L/r1H	Track/Sektor-Adresse für Block.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$02=Ungültige Block-Adresse (INV_BLOCK).
	r4	Word, Zeiger auf diskBlkBuf.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
Unverändert:	r1L/r1H	Track/Sektor-Adresse für Block.
Aufruf von:	GetBlock	

K.12.3.9 PutDiskBlkBlock (\$903f)

Einzelnen Block in *diskBlkBuf* auf Diskette schreiben.

Übergabe:	r1L/r1H	Track/Sektor-Adresse für Block.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$02=Ungültige Block-Adresse (INV_BLOCK).
	r4	Word, Zeiger auf diskBlkBuf.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
Unverändert:	r1L/r1H	Track/Sektor-Adresse für Block.
Aufruf von:	PutBlock	

K.12.3.10 GetOPDPtr (\$9036)

Adresse des Borderblock einer Diskette einlesen.

Übergabe:	n/a	-
Rückgabe:	x y r1L/r1H	Fehlerstatus, \$00=Kein Fehler \$ff = GEOS-Diskette, \$00 = Keine GEOS-Diskette Track-/Sektoradresse des Borderblock.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y, r4, r5 isGEOS	\$ff = GEOS-Diskette, \$00 = Keine GEOS-Diskette
Aufruf von:	GetDirHead ChkDkGEOS	
Hinweis:	Nur MP3: Auch als GetBorderBlock bezeichnet. "The Hitchhikers Guide to GEOS": Die Routine wird hier als GetOffPageTrSc bezeichnet.	

K.12.3.11 AccessCache (\$c2ef; C128)

Zugriff auf den Verzeichnis-Cache bei 1541- und 1571-Laufwerken.

Übergabe:	r1H r4 y	Sektor-Adresse Verzeichnisblock (0-18). Word, Zeiger auf Datenpuffer (z.B. diskBlkBuf). Zugriffs-Modus: \$ff: Cache löschen/initialisieren. Schreibt \$00,\$00 in die ersten beiden Bytes aller Sektoren im Cache (BackRAM von \$ac00-\$bfff). %0000:0000: Block in Cache speichern. %0000:0001: Block aus Cache einlesen. Wenn die ersten beiden Bytes \$00,\$00 enthalten, dann Block nicht im Cache gespeichert. %0000:0010: Block mit Cache tauschen. %0000:0011: Block mit Cache vergleichen.
Rückgabe:		Nur bei Zugriffs-Modus Block/Cache vergleichen: a / y Zero-Flag = 0 Zero-Flag = 1 Ergebnis von DoBOP: \$ff: Block und Cache unterschiedlich. \$00: Block und Cache identisch.
Verwendet:	n/a	
Verändert:	a, x, y	
Unverändert:	r1H, r4	
Aufruf von:	DoBOP	
Hinweis:	Nur MP3-128: Wird von den Laufwerkstreibern nicht mehr unterstützt. Nur GEOS V2: Die Routine wird über nur über den Laufwerkstreiber aufgerufen, nicht durch Programme.	

K.12.3.12 GetBAMBlock (\$9056; MP3)

Liest einen BAM-Block nach *dir2Head* ein.

Übergabe:	a	Sektor-Adresse des BAM-Blocks.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$40=Ungültiger Befehl (ILLEGAL DEVICE).
Verwendet:	curDrive curDirHead dir2Head dir3Head	Aktuelles Laufwerk. Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Verändert:	a, y, r1, r4	
Verwendet von:	GetDirHead	
Aufruf von	GetBlock oder ReadBlock	Abhängig davon ob InitForIO aktiv ist oder nicht.
	PutBAMBlock	Wenn aktueller BAM-Sektor verändert wurde.
Hinweis:		Nur NativeMode-Laufwerke.

K.12.3.13 PutBAMBlock (\$9059; MP3)

Schreibt einen BAM-Block in *dir2Head* auf Diskette.

Übergabe:	a	Sektor-Adresse des BAM-Blocks.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$40=Ungültiger Befehl (ILLEGAL DEVICE).
Verwendet:	curDrive curDirHead dir2Head dir3Head	Aktuelles Laufwerk. Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Verändert:	a, y, r1, r4	
Verwendet von:	PutDirHead	
Aufruf von	PutBlock oder WriteBlock	Abhängig davon ob InitForIO aktiv ist oder nicht.
Hinweis:		Nur NativeMode-Laufwerke.

K.12.3.14 SendFloppyCom (\$906b; MP3)

Befehl an Disketten-Laufwerk senden.

Übergabe:	r0 r2L	Word, Zeiger auf Floppy-Befehl. Byte, Länge des Befehls.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$40=Ungültiger Befehl (ILLEGAL DEVICE).
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y	
Unverändert:	r0, r2L	
Erfordert:	ExitTurbo InitForIO DoneWithIO	
Hinweis:		Die Länge des Floppy-Befehls ist begrenzt, die 1541 unterstützt z.B. nur 40 Zeichen!



ACHTUNG! Die folgenden Routinen gelten nur für GEOS/MegaPatch-Laufwerkstreiber, die über die Formatkennung "DDX" (*DiskDrvExtType*, \$9074) verfügen, siehe **Teil D, Kapitel 2.2 ab Seite 466**.

K.12.3.15 InitForDDrvOp (\$907c; MP3)

Zeiger auf Laufwerkstreiber im RAM und in Bank#0 der REU setzen.

Übergabe:	n/a	
Rückgabe:	r0 r1 r2 r3L	Word, Zeiger auf Laufwerkstreiber im RAM. Word, Zeiger auf Laufwerkstreiber in REU, Bank #0. Word, Größe Laufwerkstreiber. Byte, Speicherbank in REU (immer 0).
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, x, y	
Hinweis:		Die Register r0 bis r3L dürfen nach Aufruf der Routine bis DoneWithDDrvOp nicht verändert werden.

K.12.3.16 DoneWithDDrvOp (\$907f; MP3)

Zeiger auf Laufwerkstreiber im RAM und in Bank#0 der REU setzen.

Übergabe:	n/a	
Rückgabe:	n/a	
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, x, y, r0 bis r3L	
Hinweis:		Setzt die Register r0L bis r3L auf die Werte vor dem Aufruf von InitForDDrvOp zurück.

K.12.4 Die VLIR-Dateiroutinen

K.12.4.1 OpenRecordFile (\$c274)

Öffnet eine VLIR-Datei.

Übergabe:	r0	Word, Zeiger auf Dateiname.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$0a=Keine VLIR-Datei (STRUCT_MISMATCH).
	r1L/r1H	Track-/Sektoradresse des VLIR-Indexblock.
	r5 dirEntryBuf	Zeiger auf Eintrag in Verzeichnisblock. Verzeichniseintrag für Datei.
	usedRecords curRecord fileWritten	Anzahl Records in VLIR-Datei. >\$00: Datensatz-Nr., \$ff=Kein Datensatz vorhanden. \$00: Datei noch nicht geändert.
Verwendet:	curDrive	Aktuelles Laufwerk.
Verändert:	a, y, r4 bis r6 diskBlkBuf fileHeader	VLIR-Indexblock.
Aufruf von:	FindFile	

K.12.4.2 CloseRecordFile (\$c277)

Aktuell geöffnete VLIR-Datei schließen.

Übergabe:	n/a	
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
Verwendet:	curDrive fileHeader	Aktuelles Laufwerk. VLIR-Indexblock.
Verändert:	a, y, r1, r4, r5 fileWritten fileSize	Wird auf FALSE (\$00) gesetzt. Aktuelle Dateigröße.
	diskBlkBuf curDirHead dir2Head dir3Head	Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
		Nach Angabe im "Official GEOS Reference Guide" und "Hitchhikers Guide to GEOS", allerdings nur wenn seit OpenRecordFile keine anderen Routinen diesen Bereich verändert haben:
	dirEntryBuf	Verzeichniseintrag für Datei.
Aufruf von:	UpdateRecordFile	

K.12.4.3 UpdateRecordFile (\$c295)

Geöffnete VLIR-Datei aktualisieren.

Übergabe:	n/a	
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
Verwendet:	curDrive fileHeader fileWritten year, month, day hour, minutes	Aktuelles Laufwerk. VLIR-Indexblock. Wenn FALSE (\$00), dann wurde die Datei nicht verändert und muss nicht aktualisiert werden. Zum aktualisieren des Datums der letzten Änderung im Verzeichnis-Eintrag.
Verändert:	a, y, r1, r4, r5 fileWritten fileSize diskBlkBuf curDirHead dir2Head dir3Head	Wird auf FALSE (\$00) gesetzt. Aktuelle Dateigröße. Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Aufruf von:	GetDiskBlkBuf PutDiskBlkBuf PutDirHead	

K.12.4.4 PointRecord (\$c280)

Datensatz zur Bearbeitung auswählen.

Übergabe:	a	Datensatznummer (0-126).
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$08=Datensatz nicht verfügbar (INV_RECORD).
	a	Wenn kein Fehler aufgetreten ist: Datensatznummer.
	y	Track-Adresse des Datensatzes, \$00=Keine Daten.
	r1L/r1H	Track-/Sektoradresse des ersten Block im Datensatz: \$00,\$ff: Keine Daten im Datensatz.
Verwendet:	curDrive usedRecords fileHeader	Aktuelles Laufwerk. Anzahl verfügbare Datensätze. VLIR-Indexblock.
Verändert:	curRecord	Neue Datensatznummer.

K.12.4.5 NextRecord (\$c27a)

Zeiger auf nächsten Datensatz der VLIR-Datei setzen.

Übergabe:	n/a	
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$08=Datensatz nicht verfügbar (INV_RECORD).
	a	Neue Datensatznummer.
	y	Track-Adresse des Datensatzes, \$00=Keine Daten.
	r1L/r1H	Track-/Sektoradresse des ersten Block im Datensatz: \$00,\$ff: Keine Daten im Datensatz.
Verwendet:	curDrive	Aktuelles GEOS-Laufwerk.
	curRecord	Aktueller Datensatz.
	usedRecords	Anzahl verfügbare Datensätze.
	fileHeader	VLIR-Indexblock.
Verändert:	curRecord	Neue Datensatznummer.
Aufruf von:	PointRecord	

K.12.4.6 PreviousRecord (\$c27d)

Zeiger auf vorherigen Datensatz der VLIR-Datei setzen.

Übergabe:	n/a	
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$08=Datensatz nicht verfügbar (INV_RECORD).
	a	Neue Datensatznummer.
	y	Track-Adresse des Datensatzes, \$00=Keine Daten.
	r1L/r1H	Track-/Sektoradresse des ersten Block im Datensatz: \$00,\$ff: Keine Daten im Datensatz.
Verwendet:	curDrive	Aktuelles GEOS-Laufwerk.
	curRecord	Aktueller Datensatz.
	usedRecords	Anzahl verfügbare Datensätze.
	fileHeader	VLIR-Indexblock.
Verändert:	curRecord	Neue Datensatznummer.
Aufruf von:	PointRecord	

K.12.4.7 AppendRecord (\$c289)

Neuen Datensatz nach dem aktuellen Datensatz einfügen.

Übergabe:	n/a	
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$08=Datensatz nicht verfügbar (INV_RECORD).
Verwendet:	curDrive curRecord fileHeader fileWritten	Aktuelles GEOS-Laufwerk. Aktueller Datensatz. VLIR-Indexblock. TRUE (\$ff) = Datei geändert.
Verändert:	a, y, r1, r4 usedRecords fileWritten curDirHead dir2Head dir3Head	Anzahl verfügbare Datensätze. Wird auf TRUE (\$ff) gesetzt. Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Aufruf von:	GetDirHead	War fileWritten zuvor FALSE (\$00), dann wird über GetDirHead die BAM im Speicher aktualisiert.

K.12.4.8 InsertRecord (\$c286)

Neuen Datensatz vor dem aktuellen Datensatz einfügen.

Übergabe:	n/a	
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$08=Datensatz nicht verfügbar (INV_RECORD).
Verwendet:	curDrive curRecord fileHeader fileWritten	Aktuelles GEOS-Laufwerk. Aktueller Datensatz. VLIR-Indexblock. TRUE (\$ff) = Datei geändert.
Verändert:	a, y, r1, r4 usedRecords fileWritten curDirHead dir2Head dir3Head	Anzahl verfügbare Datensätze. Wird auf TRUE (\$ff) gesetzt. Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Aufruf von:	GetDirHead	War fileWritten zuvor FALSE (\$00), dann wird über GetDirHead die BAM im Speicher aktualisiert.

K.12.4.9 DeleteRecord (\$c283)

Aktuellen Datensatz aus dem VLIR-Indexblock löschen.

Übergabe:	n/a	
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$08=Datensatz nicht verfügbar (INV_RECORD).
Verwendet:	curDrive curRecord fileHeader fileWritten	Aktuelles GEOS-Laufwerk. Aktueller Datensatz. VLIR-Indexblock. TRUE (\$ff) = Datei geändert.
Verändert:	a, y, r0 bis r9 usedRecords fileWritten fileSize diskBlkBuf curDirHead dir2Head dir3Head	Anzahl verfügbare Datensätze. Wird auf TRUE (\$ff) gesetzt. Aktuelle Dateigröße. Alle Laufwerke. 1571, 1581, NativeMode. 1581, NativeMode.
Aufruf von:	GetDirHead	War fileWritten zuvor FALSE (\$00), dann wird über GetDirHead die BAM im Speicher aktualisiert.

K.12.4.10 ReadRecord (\$c28c)

Aktuellen Datensatz in den Speicher lesen.

Übergabe:	r7 r2	Word, Zeiger auf Datenpuffer. Word, Größe des Datenpuffer in Bytes.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler. \$08=Datensatz nicht verfügbar (INV_RECORD). \$0b=Pufferüberlauf (BFR_OVERFLOW).
	fileTrScTab r2 r7	Tabelle mit gelesenen Track-/Sektoradressen. Word, Nicht genutzte Bytes im Datenpuffer. Word, Zeiger hinter das zuletzt gelesene Byte.
	r1L/r1H	Bei einem Pufferüberlauf: Track-/Sektoradresse des zuletzt gelesenen Block.
Verwendet:	curDrive curRecord fileHeader	Aktuelles GEOS-Laufwerk. Aktueller Datensatz. VLIR-Indexblock.
Verändert:	a, y, r3, r4 diskBlkBuf	
Aufruf von:	ReadFile	ReadFile kann max. 32.258 Bytes einlesen, da fileTrScTab max. 127 Blocks aufnehmen kann.

K.12.4.11 WriteRecord (\$c28f)

Aktuellen Datensatz im Speicher auf Disk schreiben.

Übergabe:	r7	Word, Zeiger auf Datenpuffer.
	r2	Word, Größe des Datenpuffer in Bytes.
Rückgabe:	x	Fehlerstatus, \$00=Kein Fehler.
	fileTrScTab	Tabelle mit den Track-/Sektoradressen der auf Disk geschriebenen Blöcke.
Verwendet:	curDrive	Aktuelles GEOS-Laufwerk.
	curRecord	Aktueller Datensatz.
	fileHeader	VLIR-Indexblock.
	fileWritten	FALSE (\$00) = Datei nicht geändert.
Verändert:	a, y, r0 bis r9	Wird auf TRUE (\$ff) gesetzt.
	fileWritten	Aktuelle Dateigröße.
	fileSize	
	diskBlkBuf	Alle Laufwerke.
	curDirHead	1571, 1581, NativeMode.
	dir2Head	1581, NativeMode.
Aufruf von:	GetDirHead	War fileWritten zuvor FALSE (\$00), dann wird über GetDirHead die BAM im Speicher aktualisiert.
	BlkAlloc	Speicher für Datensatz reservieren.
	WriteFile	Speicher auf Diskette schreiben: WriteFile kann max. 32.258 Bytes schreiben, da fileTrScTab max. 127 Blocks aufnehmen kann.
Hinweis:		BAM wird erst durch UpdateRecordFile oder CloseRecordFile auf Disk aktualisiert!

K.13 Druckroutinen

GEOS-Routinen für den Zugriff auf den Drucker.

K.13.1 InitForPrint (\$7900)

Druckertreiber vor dem Druckvorgang einmalig initialisieren.

Übergabe: n/a

Rückgabe: n/a

Verändert: a, x, y, r0 bis r15

Hinweis: Im x-Register wird kein Fehler übergeben, einige Druckertreiber beenden die Routine direkt mit rts.

K.13.2 StartPrint (\$7903)

Drucker und seriellen Bus für die Übertragung von Grafikdaten vorbereiten.

Übergabe: r1 Word, Zeiger auf einen 1920-Byte-Arbeitspuffer.

Rückgabe: x Fehlerstatus, \$00=Kein Fehler.

Verändert: a, y, r0 bis r15

K.13.3 PrintBuffer (\$7906)

Grafikdaten an Drucker senden.

Übergabe: r0 Word, Zeiger auf Puffer mit Grafikdaten (640 Bytes).
r1 Word, Zeiger auf einen 1920-Byte-Arbeitspuffer.
r2 Word, Zeiger auf Puffer mit Farbdaten (80 Bytes).

Rückgabe: n/a

Verändert: a, x, y, r0 bis r15

K.13.4 StopPrint (\$7909)

Ausdruck beenden.

Übergabe: r0 Word, Zeiger auf Puffer mit Grafikdaten (640 Bytes).
r1 Kann auch auf \$0000 gesetzt werden.
Word, Zeiger auf einen 1920-Byte-Arbeitspuffer.

Rückgabe: n/a

Verändert: a, x, y, r0 bis r15

K.13.5 GetDimensions (\$790c)

Größe des Druckbereichs abfragen.

Übergabe: n/a

Rückgabe: x Seitenbreite in Cards.
y Anzahl Druckzeilen in Cards.

Verändert: n/a

K.13.6 StartASCII (\$7912)

Drucker und seriellen Bus für die Übertragung von Textdaten vorbereiten.

Übergabe: r1 Word, Zeiger auf einen 640-Byte-Arbeitspuffer.

Rückgabe: x Fehlerstatus, \$00=Kein Fehler.

Verändert: a, y, r0 bis r15

K.13.7 PrintASCII (\$790f)

Textdaten im ASCII-Format (32-126) an Drucker senden.

Übergabe: r0 Word, Zeiger auf Puffer mit Textdaten.

r1 Word, Zeiger auf einen 640-Byte-Arbeitspuffer.

Rückgabe: n/a

Verändert: a, x, y, r0 bis r15

K.13.8 SetNLQ (\$7915)

Schönschriftmodus (NLQ) im Drucker aktivieren.

Übergabe: n/a

Rückgabe: n/a

Verändert: a, x, y, r0 bis r15

K.13.9 PrintFcodes (\$7918)

Druckbefehle (inkl. NULL-Bytes) an Drucker senden.

Übergabe: r0 Word, Zeiger auf Druckbefehl.

r1 Word, Zeiger auf einen 1920-Byte-Arbeitspuffer.

r2 Word, Länge des Druckbefehls (in Bytes).

Rückgabe: x Fehlerstatus, \$00=Kein Fehler.

Verändert: a, y, r0 bis r15

Hinweis: Wird nicht von allen Druckertreibern unterstützt!

Bei Verwendung von "PrintText", einer speziellen Drucksoftware für GEOS, kann es evtl. erforderlich sein Befehle an den Drucker zu senden, z.B. um die Farbe zu wechseln. Bei anderen Druckertreibern kann der Aufruf zum Absturz führen.

K.14 Die restlichen Routinen

Sonstige GEOS-Systemroutinen.

K.14.1 MainLoop (\$c1c3)

Die MainLoop steuert den Maus/Tastatur, Menüs und führt Prozesse aus.

Übergabe: n/a

Rückgabe: n/a

Verändert: a, x, y, r0 bis r15

Hinweis: Zur internen Verwendung in GEOS, nicht zum Aufruf durch Application geeignet.

K.14.2 InterruptMain (\$c100)

Der Interrupt steuert die Mauseingabe, Prozesse und Zufallszahlen.

Übergabe: n/a

Rückgabe: n/a

Verändert: a, x, y, r0 bis r15

Aufruf von: GetRandom

Hinweis: Zur internen Verwendung in GEOS, nicht zum Aufruf durch Application geeignet.

K.14.3 GetRandom (\$c187)

Erzeugt bei jedem Aufruf eine neue 16-Bit-Zufallszahl.

Übergabe: n/a

Rückgabe: random Word, 16-Bit-Zufallszahl.

Verändert: a

K.14.4 CallRoutine (\$c1d8)

Aufruf einer Sub-Routine wenn die Adresse ungleich \$0000 ist.

Übergabe: a Lowbyte der Adresse
x Highbyte der Adresse

Rückgabe: n/a

Verändert: n/a

K.14.5 DoInlineReturn (\$c2a4)

Rücksprung aus einer selbst erstellten Inline-Routine zur Hauptanwendung.

Übergabe: a Anzahl Inline-Bytes +1.
 Statusregister Übergabe im obersten Byte auf dem Stack.
 returnAddress Word, Rücksprungadresse -1.

Rückgabe: n/a

Verwendet: n/a

Verändert: a

Hinweis: Aufruf über Befehl jmp, nicht jsr!

Beispiel für DoInlineReturn:

```

        jsr    i_Testroutine    ; Inline-Routine aufrufen
        w      $0000           ; 3 Byte Inline-Daten
        b      $00
        ...                    ; Weiter im Programm

:i_Testroutine
        pla           ; Rücksprungadresse -1 retten
        sta    returnAddress +0
        pla
        sta    returnAddress +1

        ...                    ; Inline-Routine ausführen

        php           ; Prozessorstatus auf Stack ablegen

        lda    #3 +1         ; 3 Byte Inline-Daten, 1 Byte für
                                ; Korrektur der Rücksprungadresse

        jmp     DoInlineReturn ; Inline-Routine beenden

```

K.14.6 EnterDeskTop (\$c22c)

Application beenden und zum DeskTop zurückkehren.

Übergabe: n/a

Rückgabe: n/a Keine Rückkehr zur Anwendung.

Aufruf von: SetDevice GEOS V2: Suche nach "DESK TOP" (GEOS64) oder
 OpenDisk "128 DESKTOP" (GEOS128).
 GetFile

StartAppl

Hinweis: GEOS V2: Es wird entweder auf Laufwerk 8/9 oder auf
 Laufwerk 10/11 nach DeskTop gesucht.

Nur MP3: Es wird zuerst auf RAM-Laufwerken 8 bis 11 nach
 DeskTop gesucht, danach auf allen physischen Laufwerken
 von 8 bis 11.

K.14.7 Panic (\$c2c2)

Zeigt die "Systemfehler nach \$xxxx"-Dialogbox an.

Übergabe: n/a

Rückgabe: n/a

Keine Rückkehr zur Anwendung.

Hinweis

Nur GEOS128: Zur Berechnung der Absturzadresse werden 8 zusätzliche Bytes vom Stack geholt.
Die darauf folgenden beiden Byte definieren dann die eigentliche Absturzadresse.

Nur MP3: Nach Klick auf »OK« wird über die Routine *EnterDeskTop* versucht zum DeskTop zurückzukehren.

Bei einem Programmfehler kann allerdings der GEOS-Kernal beschädigt worden sein und GEOS muss neu gestartet werden.

Beispiel für Debug-Funktion (nur MP3-64):

```
::40a0      ...      ; Programm-Code ab Adresse $40a0
::40a3      jsr      Panic      ; Wird diese Routine ausgeführt?
          ...
```

Beispiel für Debug-Funktion (nur MP3-128):

```
::40a0      ...      ; Programm-Code ab Adresse $40a0
::40a3      jsr      Panic128   ; Wird diese Routine ausgeführt?
          ...

:Panic128   ldx      #8          ; 8 Dummy-Bytes auf Stack ablegen
::1         pha
          dex
          bne      :1
          jmp      Panic      ; Systemfehler anzeigen
```

In beiden Fällen erscheint die Panic-Dialogbox mit der Adresse \$40a3. Mit »OK« kehrt man über *EnterDeskTop* zum DeskTop zurück.

K.14.8 BootGeos (\$c000)

Neustart von GEOS aus BASIC heraus.

Übergabe: n/a

Hinweis:

Nur GEOS V2: Funktioniert nur, wenn der Speicher von \$c000-\$c07f nicht verändert wurde. Ist keine REU vorhanden, wird GEOS von Laufwerk 8 geladen.

MP3-64: Funktioniert nur, wenn der Speicher von \$c000 bis \$c0ff nicht verändert wurde.

Hinweis: GEOS/MP3-64 bis Version 3.3r10, GEOS/MP3 mit CMD-RAMLink und/oder CMD-SuperCPU funktionieren nicht, wenn der Speicher von \$9d80 bis \$9fff verändert wurde.

MP3-128: Funktioniert nur, wenn der Speicher von \$c000 bis \$c0ff und \$9d80 bis \$9fff in Bank 1 nicht verändert wurde.

K.14.9 ToBasic (\$c241)

GEOS beenden und zum BASIC-Modus wechseln.

Übergabe:	r0	Zeiger auf BASIC-Befehl. Soll kein Befehl ausgeführt werden, muss r0 auf ein NULL-Byte zeigen.
	r5	Kein Programm nachladen: \$0000.
	r7	C64: \$0803, C128: \$1c03.
	r5	Programm ist bereits geladen: \$0000.
	r7	Word, Zeiger auf erstes Byte hinter dem Programm.
	r5	Programm nachladen: Word, Zeiger auf Zeichniseintrag. (z.B. dirEntryBuf)
	r7	Ladeadresse für das Programm.
Rückgabe:	n/a	Keine Rückkehr zur Anwendung.

K.14.10 CRC (\$c20e)

Erzeugt eine Prüfsumme über einen vorgegebenen Speicherbereich.

Übergabe:	r0	Word, Zeiger auf Speicherbereich.
	r1	Word, Größe des Speicherbereichs.
Rückgabe:	r2	Word, CRC-Prüfsumme.
Verändert:	a, x, y, r0, r1, r3L	

K.14.11 GetSerialNumber (\$c196)

Übergibt die Seriennummer des GEOS-Systems.

Übergabe:	n/a	
Rückgabe:	r0	Word, GEOS-Seriennummer, 16-Bit-Wert
Verändert:	a	

K.14.12 GEOS_InitSystem (\$c0ee; MP3)

Setzt alle GEOS- und I/O-Register auf Standardwerte zurück.

Übergabe:	n/a
Rückgabe:	n/a
Verwendet:	n/a
Verändert:	a, x, y, r0 bis r2L mouseTop mouseBottom mouseLeft mouseRight

Hinweis: Vor dem Aufruf muss der Interrupt über php und sei gesperrt und anschließend über plp wieder freigegeben werden!

K.14.13 PutKeyInBuffer (\$c0f1; MP3)

Speichert einen Tastencode / Neue Taste im Tastaturpuffer.

Übergabe: a Tastencode.
Rückgabe: pressFlag Bit 7=1: Neue Taste im Tastaturpuffer.
Verändert: a, x, y

K.14.14 SCPU_Pause (\$c0f4; MP3)

Führt eine Pause von ca. 1/10 Sek. aus.

Übergabe: n/a
Rückgabe: n/a
Verändert: a

Hinweis: Funktioniert auch bei SuperCPU oder anderen Beschleunigerkarten.
 Funktioniert nicht im VICE-Emulator im Warp-Modus, da hier auch der Timer beschleunigt wird.

K.14.15 SCPU_OptOn (\$c0f7; MP3)

Aktiviert die Optimierung für GEOS und die CMD SuperCPU.

Übergabe: n/a
Rückgabe: n/a
Verändert: a, x, y
 Flag_Optimize \$00: Optimierung aktiviert.

Hinweis: Spiegelt VIC-Bank#2 (\$8000-\$bfff) im RAM der SuperCPU für schnelleren RAM-Zugriff.

K.14.16 SCPU_OptOff (\$c0fa; MP3)

Deaktiviert die Optimierung für GEOS und die CMD SuperCPU.

Übergabe: n/a
Rückgabe: n/a
Verändert: a, x, y
 Flag_Optimize \$03: Optimierung deaktiviert.

Hinweis: Notwendig für Zugriff auf Textbildschirm.

K.14.17 SCPU_SetOpt (\$c0fd; MP3)

Deaktiviert die Optimierung für GEOS und die CMD SuperCPU.

Übergabe: Flag_Optimize Byte, \$00: Optimierung aktivieren.
 Byte, \$03: Optimierung deaktivieren.
Rückgabe: n/a
Verändert: a, x, y

K.15 Das Registermenü von GEOS/MegaPatch

Routinen für die Arbeit mit Registermenüs.

K.15.1 DoRegister (\$6d00; MP3)

Zeichnet und aktiviert das Registermenü.

Übergabe: r0 Word, Zeiger auf Menü-Tabelle.

Rückgabe: n/a

Verändert: a, x, y, r0 bis r15

Hinweis: Während des Aufbaus des Registermenüs darf das Register r15 nicht verändert werden.

K.15.2 ExitRegisterMenu (\$6d03; MP3)

Deaktiviert die Routinen für das Registermenü.

Übergabe: n/a

Rückgabe: n/a

Verändert: a, x, y, r0

K.15.3 RegisterInitMenu (\$6d06; MP3)

Zeichnet das aktuelle Register erneut auf den Bildschirm.

Übergabe: n/a

Rückgabe: n/a

Verändert: a, x, y, r0 bis r15

K.15.4 RegisterUpdate (\$6d09; MP3)

Aktualisiert einen Eintrag auf der aktuellen Registerseite.

Übergabe: r15 Word, Zeiger auf einen 11-Byte-Registereintrag.

Rückgabe: n/a

Verändert: a, x, y, r0 bis r15

K.15.5 RegisterAllOpt (\$6d0c; MP3)

Registerseite löschen, Inhalte neu zeichnen.

Übergabe: n/a

Rückgabe: n/a

Verändert: a, x, y, r0 bis r15

K.15.6 RegisterNextOpt (\$6d0f; MP3)

Aktualisiert alle Inhalte der Registerseite ohne die Seite zu löschen.

Übergabe: n/a

Rückgabe: n/a

Verändert: a, x, y, r0 bis r15

K.15.7 RegDrawOptFrame (\$6d12; MP3)

Zeichnet einen um 1-Pixel größeren Rahmen um ein Rechteck.

Übergabe:	r2L	Byte, y-Koordinate, obere Seite des Rechtecks.
	r2H	Byte, y-Koordinate, untere Seite des Rechtecks.
	r3	Word, x-Koordinate, linke Seite des Rechtecks.
	r4	Word, x-Koordinate, rechte Seite des Rechtecks.
Verwendet:	dispBufferOn	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich.
Verändert:	a, x, y	Bereich um 1 Pixel in alle Richtungen vergrößert.
	r2L, r2H, r3, r4	
	r5 bis r9, r11	

K.15.8 RegClrOptFrame (\$6d15; MP3)

Löscht einen um 1-Pixel größeren Rahmen um ein Rechteck.

Übergabe:	r2L	Byte, y-Koordinate, obere Seite des Rechtecks.
	r2H	Byte, y-Koordinate, untere Seite des Rechtecks.
	r3	Word, x-Koordinate, linke Seite des Rechtecks.
	r4	Word, x-Koordinate, rechte Seite des Rechtecks.
Verwendet:	dispBufferOn	Bit 7=1: In Vordergrund schreiben. Bit 6=1: In Hintergrund schreiben. Kombination von Bit 7 und Bit 6 möglich.
Verändert:	a, x, y	Bereich um 1 Pixel in alle Richtungen vergrößert.
	r2L, r2H, r3, r4	
	r5 bis r9, r11	

K.15.9 RegisterSetFont (\$6d18; MP3)

Aktiviert den Zeichensatz für das Registermenü.

Übergabe:	n/a	
Rückgabe:	baselineOffset	Position der Grundlinie im Register-Zeichensatz.
	curSetWidth	Länge einer Bitstream-Reihe in Bytes.
	curHeight	Anzahl Bitstream-Zeilen im Zeichensatz.
	curIndexTable	Zeiger auf aktuelle Index-Tabelle.
	cardDataPtr	Zeiger auf erste Bitstream-Reihe.
Aufruf von:	LoadCharSet	
Verändert:	a, y, r0	

Anhang L

L.1 Die C64-Tastaturmatrix

Über die beiden CIA-Register in \$dc00 und \$dc01 kann die Tastatur des C64 direkt abgefragt werden. Dazu setzt man in \$dc00 alle Bit auf den Wert 1 mit Ausnahme der Spalte, in welcher sich die gesuchte Taste befindet. Anschließend liest man aus \$dc01 den Tastenstatus aus. Nicht gedrückte Tasten haben in der Zeile das entsprechend Bit gesetzt, nur das Bit für die gedrückte Taste hat den Wert 0.

Hier nun die Tastaturmatrix des C64:

Register \$dc00									
Register \$dc01		b7	b6	b5	b4	b3	b2	b1	b0
	b0	1	£	+	9	7	5	3	DEL
	b1	◀	*	P	I	Y	R	W	RETURN
	b2	CTRL	;	L	J	G	D	A	CRSR-R
	b3	2	HOME	-	0	8	6	4	F/
	b4	SPACE	R-SHIFT	.	M	B	C	Z	F1
	b5	CBM	=	:	K	H	F	S	F3
	b6	Q	▲	@	O	U	T	E	F5
	b7	STOP	/	,	N	V	X	L-SHIFT	CRSR-D

Hier ein Beispiel um die Taste **[CBM]** abzufragen:

```
:wait      php                      ; Interrupt sperren
           sei
           ldx      CPU_DATA        ; Speicher-Register
           ldx      #IO_IN          ; I/O-Speicherbereich einblenden
           stx      CPU_DATA
           lda      #%0111 1111     ; b7 in $dc00 löschen
           sta      $dc00
           lda      $dc01           ; Tastenstatus aus $dc01 auslesen
           stx      CPU_DATA        ; Speicher-Register zurücksetzen
           plp

           and      #%0010 0000     ; b5 in $dc01 gelöscht?
           bne      wait            ; Nein, warten...
```

In den letzten beiden Zeilen wird getestet, ob die Taste **[CBM]** alleine gedrückt wurde, alternativ kann man aber mit einer kleinen Änderung auch testen ob mehrere Tasten gedrückt wurden. Hier ein Beispiel für eine Abfrage von **[CBM]+[0]**:

```
...
lda      #%0110 1111             ; b7="CBM", b4="0" in $dc00 löschen
sta      $dc00
lda      $dc01                   ; Tastenstatus aus $dc01 auslesen
...
cmp      #%1101 0111             ; Nur b5+b3, beide Bit gelöscht?
bne      wait                     ; Nein, warten
```

L.2 Die Tastaturcodes von GEOS

Im **Teil C, Anhang I.1 auf Seite 417** ist die Tastaturbelegung bereits beschrieben worden. Für die Auswertung von *keyData* folgt eine tabellarische Übersicht, die aufgeführten Tastencodes gelten dabei für die deutsche GEOS-Version:

Hex	Taste	Hex	Taste	Hex	Taste	Hex	Taste
\$00	n.v.	\$20	SPACE	\$40	3 +SHIFT §	\$60	- +SHIFT `
\$01	F1	\$21	1 +SHIFT !	\$41	A +SHIFT A	\$61	A a
\$02	F2	\$22	2 +SHIFT "	\$42	B +SHIFT B	\$62	B b
\$03	F3	\$23	= #	\$43	C +SHIFT C	\$63	C c
\$04	F4	\$24	4 +SHIFT \$	\$44	D +SHIFT D	\$64	D d
\$05	F5	\$25	5 +SHIFT %	\$45	E +SHIFT E	\$65	E e
\$06	F6	\$26	6 +SHIFT &	\$46	F +SHIFT F	\$66	F f
\$07	G +CTRL	\$27	- .	\$47	G +SHIFT G	\$67	G g
\$08	CRSR-L	\$28	8 +SHIFT (\$48	H +SHIFT H	\$68	H h
\$09	I +CTRL	\$29	9 +SHIFT)	\$49	I +SHIFT I	\$69	I i
\$0a	J +CTRL	\$2a	* +SHIFT *	\$4a	J +SHIFT J	\$6a	J j
\$0b	K +CTRL	\$2b	* +	\$4b	K +SHIFT K	\$6b	K k
\$0c	L +CTRL	\$2c	, /	\$4c	L +SHIFT L	\$6c	L l
\$0d	RETURN	\$2d	/ -	\$4d	M +SHIFT M	\$6d	M m
\$0e	F7	\$2e	. .	\$4e	N +SHIFT N	\$6e	N n
\$0f	F8	\$2f	7 +SHIFT /	\$4f	O +SHIFT O	\$6f	O o
\$10	CRSR-UP	\$30	0 0	\$50	P +SHIFT P	\$70	P p
\$11	CRSR-DN	\$31	1 1	\$51	Q +SHIFT Q	\$71	Q q
\$12	HOME	\$32	2 2	\$52	R +SHIFT R	\$72	R r
\$13	CLR	\$33	3 3	\$53	S +SHIFT S	\$73	S s
\$14	◀	\$34	4 4	\$54	T +SHIFT T	\$74	T t
\$15	U +CTRL	\$35	5 5	\$55	U +SHIFT U	\$75	U u
\$16	STOP	\$36	6 6	\$56	V +SHIFT V	\$76	V v
\$17	RUN	\$37	7 7	\$57	W +SHIFT W	\$77	W w
\$18	X +CTRL	\$38	8 8	\$58	X +SHIFT X	\$78	X x
\$19	Z +CTRL	\$39	9 9	\$59	Z +SHIFT Y	\$79	Z y
\$1a	Y +CTRL	\$3a	. +SHIFT :	\$5a	Y +SHIFT Z	\$7a	Y z
\$1b	n.v.	\$3b	, +SHIFT ;	\$5b	; +SHIFT Ä	\$7b	; ä
\$1c	INSERT	\$3c	, +CBM/S <	\$5c	: +SHIFT Ö	\$7c	: ö
\$1d	DEL	\$3d	0 +SHIFT =	\$5d	@ +SHIFT Ü	\$7d	@ ü
\$1e	CRSR-R	\$3e	. +CBM/S >	\$5e	£ +SHIFT ^	\$7e	+ ß
\$1f	n.v.	\$3f	+ +SHIFT ?	\$5f	/ +SHIFT _	\$7f	n.v.

Neben den Tasten wird noch das entsprechende BSW9/DE-Textzeichen angezeigt.

Hinweis: Das Zeichen \$27 wird auch mit **[SHIFT]+[=]** erzeugt. Tastencodes, die mit "n.v." bezeichnet sind, können unter GEOS über *keyData* nicht abgefragt werden.

Die Tasten **[£]** (ohne **[SHIFT]**) und **[▲]** (mit und ohne **[SHIFT]**) sind nicht belegt.

Hier noch die Tastencodes als Hexadezimal-Wert auf den entsprechenden Tasten:

14	31	32	33	34	35	36	37	38	39	30	7e	27		12	1d	
		71	77	65	72	74	7a	75	69	6f	70	7d	2b			
		Q	W	E	R	T	Y	U	I	O	P	@	*			
16		61	73	64	66	67	68	6a	6b	6c	7c	7b	23		0d	
		A	S	D	F	G	H	J	K	L	:	;	=			
		79	78	63	76	62	6e	6d	2c	2e	2d			11	1e	
		Z	X	C	V	B	N	M	,	.	/					
20																

01
03
05
0e

Bild K.1:
C64-Tastatur

14	21	22	40	24	25	26	2f	28	29	3d	3f	60	5e	13	1c	
		51	57	45	52	54	5a	55	49	4f	50	5d	2a			
		Q	W	E	R	T	Y	U	I	O	P	@	*			
17		41	53	44	46	47	48	4a	4b	4c	5c	5b	27		0d	
		A	S	D	F	G	H	J	K	L	:	;	=			
		59	58	43	56	42	4e	4d	3b	3a	5f			10	08	
		Z	X	C	V	B	N	M	,	.	/					
20																

02
04
06
0f

Bild K.2:
C64-Tastatur
mit [SHIFT]

94	b1	b2	b3	b4	b5	b6	b7	b8	b9	b0	fe	a7	-	92	9d	
		f1	f7	e5	f2	f4	fa	f5	e9	ef	f0	fd	ab			
		Q	W	E	R	T	Y	U	I	O	P	@	*			
96		e1	f3	e4	e6	e7	e8	ea	eb	ec	fc	fb	a3		8d	
		A	S	D	F	G	H	J	K	L	:	;	=			
		f9	f8	e3	f6	e2	ee	ed	ac	ae	ad			91	9e	
		Z	X	C	V	B	N	M	,	.	/					
a0																

81
83
85
8e

Bild K.3:
C64-Tastatur
mit [CBM]

94	a1	a2	c0	a4	a5	a6	af	a8	a9	b0	bf	5e	de	93	9c	
		d1	d7	c5	d2	d4	da	d5	c9	cf	d0	dd	aa			
		Q	W	E	R	T	Y	U	I	O	P	@	*			
17		c1	d3	c4	c6	c7	ca	cb	cc	dc	db	a7		8d		
		A	S	D	F	G	H	J	K	L	:	;	=			
		d9	d8	c3	d6	c2	ce	cd	3c	3e	df			90	88	
		Z	X	C	V	B	N	M	,	.	/					
a0																

82
84
86
8f

Bild K.4:
C64-Tastatur
mit [SHIFT] und [CBM]

14	31	32	33	34	35	36	37	38	39	30	7e	27		12	1d	
		11	17	05	12	14	1a	15	09	0f	10	7d	2b			
		Q	W	E	R	T	Y	U	I	O	P	@	*			
16		01	13	04	06	07	08	0a	0b	0c	7c	7b	23		0d	
		A	S	D	F	G	H	J	K	L	:	;	=			
		19	18	03	16	02	0e	0d	2c	2e	2d			11	1e	
		Z	X	C	V	B	N	M	,	.	/					
20																

01
03
05
0e

Bild K.5:
C64-Tastatur
mit [CTRL]

94	b1	b2	b3	b4	b5	b6	b7	b8	b9	b0	fe	a7	-	92	9d	
		91	97	85	92	94	9a	95	89	8f	90	fd	ab			
		Q	W	E	R	T	Y	U	I	O	P	@	*			
96		81	93	84	86	87	88	8a	8c	8c	fb	fb		8d		
		A	S	D	F	G	H	J	K	L	:	;	=			
		99	98	83	96	82	8e	8d	ac	ae	ad			91	9e	
		Z	X	C	V	B	N	M	,	.	/					
a0																

81
9e
85
8e

Bild K.6:
C64-Tastatur
mit [CTRL] und [CBM]

Hinweis: Einige Kombinationen sind doppelt, z.B. [RETURN] = [CTRL]+[m].

L.3 Die Tastaturcodes von GEOS/US

In den deutschen GEOS-Versionen wurden einige Tasten der deutschen Tastatur angepasst. Hier die Tastaturmatrix für *keyData* bei einem US-GEOS:

Hex	Taste	Hex	Taste	Hex	Taste	Hex	Taste
\$00	n.v.	\$20	SPACE	\$40	@ @	\$60	@ +CBM `
\$01	F1	\$21	1 +SHIFT !	\$41	A +SHIFT A	\$61	A a
\$02	F2	\$22	2 +SHIFT "	\$42	B +SHIFT B	\$62	B b
\$03	F3	\$23	3 +SHIFT #	\$43	C +SHIFT C	\$63	C c
\$04	F4	\$24	4 +SHIFT \$	\$44	D +SHIFT D	\$64	D d
\$05	F5	\$25	5 +SHIFT %	\$45	E +SHIFT E	\$65	E e
\$06	F6	\$26	6 +SHIFT &	\$46	F +SHIFT F	\$66	F f
\$07	G +CTRL	\$27	7 +SHIFT .	\$47	G +SHIFT G	\$67	G g
\$08	CRSR-L	\$28	8 +SHIFT (\$48	H +SHIFT H	\$68	H h
\$09	I +CTRL	\$29	9 +SHIFT)	\$49	I +SHIFT I	\$69	I i
\$0a	J +CTRL	\$2a	* *	\$4a	J +SHIFT J	\$6a	J j
\$0b	K +CTRL	\$2b	+ +	\$4b	K +SHIFT K	\$6b	K k
\$0c	L +CTRL	\$2c	, ,	\$4c	L +SHIFT L	\$6c	L l
\$0d	RETURN	\$2d	- -	\$4d	M +SHIFT M	\$6d	M m
\$0e	F7	\$2e	. .	\$4e	N +SHIFT N	\$6e	N n
\$0f	F8	\$2f	/ /	\$4f	O +SHIFT O	\$6f	O o
\$10	CRSR-UP	\$30	0 0	\$50	P +SHIFT P	\$70	P p
\$11	CRSR-DN	\$31	1 1	\$51	Q +SHIFT Q	\$71	Q q
\$12	HOME	\$32	2 2	\$52	R +SHIFT R	\$72	R r
\$13	CLR	\$33	3 3	\$53	S +SHIFT S	\$73	S s
\$14	◀	\$34	4 4	\$54	T +SHIFT T	\$74	T t
\$15	U +CTRL	\$35	5 5	\$55	U +SHIFT U	\$75	U u
\$16	STOP	\$36	6 6	\$56	V +SHIFT V	\$76	V v
\$17	RUN	\$37	7 7	\$57	W +SHIFT W	\$77	W w
\$18	£	\$38	8 8	\$58	X +SHIFT X	\$78	X x
\$19	Y +CTRL	\$39	9 9	\$59	Y +SHIFT Y	\$79	Y y
\$1a	Z +CTRL	\$3a	: :	\$5a	Z +SHIFT Z	\$7a	Z z
\$1b	n.v.	\$3b	; ;	\$5b	: +SHIFT [\$7b	: +CBM {
\$1c	INSERT	\$3c	, +SHIFT <	\$5c	/ +CBM \	\$7c	▲ +CBM
\$1d	DEL	\$3d	= =	\$5d	; +SHIFT]	\$7d	; +CBM }
\$1e	CRSR-R	\$3e	. +SHIFT >	\$5e	▲ ^	\$7e	* +CBM ~
\$1f	n.v.	\$3f	/ +SHIFT ?	\$5f	- +CBM _	\$7f	n.v.

Neben den Tasten noch das entsprechende BSW9/US-Textzeichen angezeigt.

Die Tastencodes, welche mit "n.v." bezeichnet sind, können unter GEOS über die Adresse *keyData* nicht abgefragt werden.

Anhang M

In diesem Abschnitt finden sich ein paar Quelltext-Beispiele.

M.1 GEOS/MegaPatch: Bildschirmschoner "Rasterbars"

Rasterbars ist ein Bildschirmschoner, der ohne erweiterten Speicher auskommt und auch die Bildschirmgrafik selbst nicht verändert. Es wird lediglich der Bildschirm abgeschaltet und die Rasterzeilen fortlaufend mit verschiedenen Farben eingefärbt.

Zu Beginn des Quelltextes kann man das Label »:BUILD_MODE« entweder auf den Wert *EN_APP_MODE* oder *DIS_APP_MODE* setzen.

Wenn der Bildschirmschoner als Anwendung gestartet werden soll, dann verwendet man *EN_APP_MODE*. Der MegaAssembler erzeugt dann eine Application, die man über den DeskTop starten kann. Das kann zum Beispiel während der Testphase hilfreich sein, weil man den Bildschirmschoner dann als Application direkt starten kann. Im Gegensatz dazu erzeugt *DIS_APP_MODE* eine Systemdatei, die vom GEOS.Editor als Bildschirmschoner verwendet werden kann.

```

;*** Build-Modus definieren:
:EN_APP_MODE   = $1000   ;Anwendung
:DIS_APP_MODE  = $2000   ;Bildschirmschoner

:BUILD_MODE    = DIS_APP_MODE

;*** Symboltabellen.
if .p
    t "TopSym"
    t "TopSym.MP3"
    t "ExtSym.MP3"
    t "MacTab"

:LANG_DE       = $0110   ;Deutsch
:LANG_EN       = $0220   ;Englisch
:LANG          = LANG_EN
endif

;*** GEOS-Header.
n "Rasterbars"
c "ScrSaver64 V1.0"
z $80          ;Bildschirmflag: nur GEOS64
o LD_ADDR_SCRSAVER

;--- GEOS-Filetyp/Startadresse
if BUILD_MODE = EN_APP_MODE
    f APPLICATION
    p APP_START
endif

if BUILD_MODE = DIS_APP_MODE
    f SYSTEM
endif

```



```

;--- Datei-Icon
i


;--- Infotext
if LANG = LANG_DE
    h "Bildschirmschoner für GDOS64..."
endif
if LANG = LANG_EN
    h "Screensaver for GDOS64..."
endif

;*** ScreenSaver aufrufen.
:MainInit    jmp    InitScrSaver

;*** ScreenSaver installieren.
;Laufwerk, von dem der ScreenSaver geladen wurde, muss noch aktiv sein!
;Rückgabe eines Installationsfehlers im xReg ($00=Kein Fehler).
;ACHTUNG!
;Nur JMP-Befehl oder "LDX #$00:RTS", da direkt im Anschluss der Name des
;ScreenSavers erwartet wird! (Adresse: LD_ADDR_SCRSAVER +6)
:InstallSaver ldx    #$00
              rts

;*** Name des ScreenSavers.
;Direkt nach dem JMP-Befehl, da über GEOS.Editor der Name an dieser Stelle
;ausgelesen wird. Der Name muss mit dem Dateinamen übereinstimmen, da der
;ScreenSaver über diesen Namen beim Systemstart geladen wird.
:SaverName    b "Rasterbars",NULL

;*** ScreenSaver als Anwendung starten.
if BUILD_MODE = EN_APP_MODE
:APP_START    lda    Flag_ScrSaver    ;ScreenSaver-Flag speichern.
              pha
              jsr    InitScrSaver    ;ScreenSaver starten.
              pla                    ;War ScreenSaver zuvor aktiv?
              bpl    :1                ;=> Ja, weiter...
              sta    Flag_ScrSaver    ;ScreenSaver wieder abschalten.
              jmp    EnterDeskTop     ;Zurück zum DeskTop.
:1
endif

;*** ScreenSaver aufrufen.
:InitScrSaver php                    ;IRQ sperren.
              sei                    ;ScreenSaver läuft in der MainLoop!

              ldx    #(r15H - r0L)    ;Register ":r0" bis ":r3"
::51          lda    r0,x              ;zwischenspeichern.
              pha
              dex
              bpl    :51

              jsr    DoSaverJob        ;Bildschirmschoner aktivieren.

```

```

        lda    #%01000000    ;Bildschirmschoner neu starten.
        sta    Flag_ScrSaver

::52    ldx    #0              ;Register ":r0" bis ":r3"
        pla                    ;zurückschreiben.
        sta    r0,x
        inx
        cpx    #(r15H - r0L) +1
        bne    :52

        sei                    ;IRQ abschalten.

        ldx    CPU_DATA      ;CPU-Register zwischenspeichern und
        lda    #IO_IN        ;I/O-Bereich einblenden.
        sta    CPU_DATA

::53    lda    #$00
        sta    $dc00          ;Tastenregister aktivieren.
        lda    $dc01          ;Tastenstatus einlesen.
        eor    #$ff           ;Taste noch gedrückt ?
        bne    :53            ;Ja, Warteschleife...

        stx    CPU_DATA      ;CPU-Register zurücksetzen.

        plp
        rts                  ;IRQ zurücksetzen und
                               ;Ende...

;*** Bildschirmschoner-Grafik.
;DoSaverJob    lda    CPU_DATA
                pha

                lda    #IO_IN    ;I/O-Bereich einblenden.
                sta    CPU_DATA

                lda    $d011      ;Bildschirm abschalten.
                and    #%11101111
                sta    $d011

                lda    $d015      ;Strite0n-Register speichern.
                pha

                lda    #$00        ;Sprites abschalten.
                sta    $d015

                lda    $d020      ;Rahmenfarbe sichern.
                pha

                lda    #$00        ;Rahmenfarbe löschen.
                sta    $d020

::51    lda    #$00              ;Warten bis keine Taste gedrückt.
        sta    $dc00
        lda    $dc01
        eor    #$ff
        bne    :51

```

```

::52      ldx    #$00          ;Zeiger auf erste Rasterzeile.
          lda    CurColTab
          tay
          clc
          adc    #$02
          cmp    #$08
          bcc    :53
          lda    #$00
::53      sta    CurColTab

          lda    ColTabVec +0,y
          sta    r0L
          lda    ColTabVec +1,y
          sta    r0H
          tya
          lsr
          tay
          lda    ColTabLen +0,y
          sta    r1L

::54      cpx    $d012        ;Rasterzeile erreicht ?
          bne    :54          ;Nein, warten.
          stx    r1H          ;Rasterzeile merken.

::55      cpx    $d012        ;Am Beginn der nächsten Zeile ?
          beq    :55          ;Nein, warten.

          ldy    #$00          ;Farbbalken erzeugen.
::56      lda    (r0L),y
          inx
::57      cpx    $d012
          beq    :57
          sta    $d020
          iny
          cpy    r1L
          bcc    :56

          lda    #$00          ;Rahmenfarbe löschen.
          sta    $d020

          dey
          dey                  ;Rasterbalken rotieren.

          lda    (r0L),y
          pha
          dey
          lda    (r0L),y
          iny
          sta    (r0L),y
          dey
          bne    :61
          pla
          sta    (r0L),y

```

```

        lda    #$00
        sta    $dc00                ;Tastenregister aktivieren.
        lda    $dc01                ;Tastenstatus einlesen.
        eor    #$ff                ;Wurde Taste gedrückt ?
        bne    :58                  ;Ja, weiter...

        ldx    r1H                  ;Zeiger auf Rasterzeile einlesen.
        inx                      ;Zeiger auf nächste Zeile.
        bne    :54
        jmp    :52                  ;Schleife...

::58      pla                      ;Rahmenfarbe zurücksetzen.
        sta    $d020
        pla                      ;Sprites wieder aktivieren.
        sta    $d015

        lda    $d011                ;Bildschirm wieder einschalten.
        ora    #%00010000
        sta    $d011

        pla
        sta    CPU_DATA            ;I/O-Bereich zurücksetzen.
        rts

;*** Farbtabellen.
;Am Anfang/Ende muss ein NULL-Byte stehen um klare Übergänge zwischen dem
;Balken und dem Bildschirmhintergrund zu erzeugen!

;--- Blau.
:ColGrfx1a    b $00
               b $06,$06,$06,$06,$06
               b $0e,$0e,$0e,$0e
               b $03,$03,$03
               b $0d,$0d
               b $01
               b $0d,$0d
               b $03,$03,$03
               b $0e,$0e,$0e,$0e
               b $06,$06,$06,$06,$06
               b $00

:ColGrfx1b

;--- Braun.
:ColGrfx2a    b $00
               b $09,$09,$09,$09,$09
               b $08,$08,$08,$08
               b $07,$07,$07
               b $0f,$0f
               b $01
               b $0f,$0f
               b $07,$07,$07
               b $08,$08,$08,$08
               b $09,$09,$09,$09,$09
               b $00

:ColGrfx2b

```

```

;--- Violett/Rot.
:ColGrfx3a      b $00
                b $06,$06,$06,$06,$06
                b $04,$04,$04,$04
                b $02,$02,$02
                b $0a,$0a
                b $01
                b $0a,$0a
                b $02,$02,$02
                b $04,$04,$04,$04
                b $06,$06,$06,$06,$06
                b $00

:ColGrfx3b

;--- Grau.
:ColGrfx4a      b $00
                b $0b,$0b,$0b,$0b
                b $0c,$0c,$0c
                b $0f,$0f
                b $01
                b $0f,$0f
                b $0c,$0c,$0c
                b $0b,$0b,$0b,$0b
                b $00

:ColGrfx4b

;*** Variablen.
:CurColTab      b $00
:ColTabVec       w ColGrfx1a
                w ColGrfx2a
                w ColGrfx3a
                w ColGrfx4a
:ColTabLen       b ColGrfx1b-ColGrfx1a
                b ColGrfx2b-ColGrfx2a
                b ColGrfx3b-ColGrfx3a
                b ColGrfx4b-ColGrfx4a

;*****
;*** Endadresse testen.
;*****
                g LD_ADDR_SCRSAVER + R2_SIZE_SCRSAVER -1
;*****

```

M.2 GEOS/MegaPatch: Bildschirmschoner "Starfield"

Dieser Bildschirmschoner speichert den aktuellen Grafikbildschirm im erweiterten GEOS-Speicher und zeichnet einen animierten Sternenhimmel.

Auch hier kann zu Beginn des Quelltextes das Label »:BUILD_MODE« entweder auf den Wert *EN_APP_MODE* oder *DIS_APP_MODE* gesetzt werden, um entweder eine Application oder eine Systemdatei zu erstellen. Einzelheiten dazu finden sich im vorherigen Abschnitt.

Der Bildschirmschoner nutzt außerdem die Kernallroutine RND um Zufallszahlen zu erzeugen. Es gibt zwar auch andere Möglichkeiten um Zufallszahlen zu erzeugen, aber damit kann die Einbindung von Kernallroutinen demonstriert werden.

```

;*** Build-Modus definieren:
;EN_APP_MODE   = $1000   ;Anwendung
;DIS_APP_MODE  = $2000   ;Bildschirmschoner

:BUILD_MODE    = DIS_APP_MODE

;*** Symboltabellen.
if .p
    t "TopSym"
    t "TopSym.MP3"
    t "ExtSym.MP3"
    t "MacTab"

:LANG_DE       = $0110   ;Deutsch
:LANG_EN       = $0220   ;Englisch
:LANG          = LANG_EN
endif

;*** GEOS-Header.
n "Starfield"
c "ScrSaver64 V1.0"
z $80          ;Bildschirmflag: nur GEOS64
o LD_ADDR_SCRSAVER

;--- GEOS-Filetyp/Startadresse
if BUILD_MODE = EN_APP_MODE
    f APPLICATION
    p APP_START
endif
if BUILD_MODE = DIS_APP_MODE
    f SYSTEM
endif

;--- Datei-Icon
i

```



```

;--- Infotext
if LANG = LANG_DE
    h "Bildschirmschoner für GDOS64..."
endif
if LANG = LANG_EN
    h "Screensaver for GDOS64..."
endif

;*** ScreenSaver aufrufen.
:MainInit    jmp    InitScrSaver

;*** ScreenSaver installieren.
;Laufwerk, von dem der ScreenSaver geladen wurde, muss noch aktiv sein!
;Rückgabe eines Installationsfehlers im xReg ($00=Kein Fehler).
;ACHTUNG!
;Nur JMP-Befehl oder "LDX #$00:RTS", da direkt im Anschluss der Name des
;ScreenSavers erwartet wird! (Adresse: LD_ADDR_SCRSAVER +6)
:InstallSaver    ldx    #$00
                rts

;*** Name des ScreenSavers.
;Direkt nach dem JMP-Befehl, da über GEOS.Editor der Name an dieser Stelle
;ausgelesen wird. Der Name muss mit dem Dateinamen übereinstimmen, da der
;ScreenSaver über diesen Namen beim Systemstart geladen wird.
:SaverName      b "Starfield",NULL

;*** ScreenSaver als Anwendung starten.
if BUILD_MODE = EN_APP_MODE
:APP_START      lda    Flag_ScrSaver    ;ScreenSaver-Flag speichern.
                pha
                jsr    InitScrSaver    ;ScreenSaver starten.
                pla                    ;War ScreenSaver zuvor aktiv?
                bpl    :1                ;=> Ja, weiter...
                sta    Flag_ScrSaver    ;ScreenSaver wieder abschalten.
:1              jmp    EnterDeskTop    ;Zurück zum DeskTop.
endif

;*** ScreenSaver aufrufen.
:InitScrSaver    php                    ;IRQ sperren.
                sei                    ;ScreenSaver läuft in der MainLoop!

::51            ldx    #(r15H - r0L)    ;Register ":r0" bis ":r3"
                lda    r0,x            ;zwischenspeichern.
                pha
                dex
                bpl    :51

                jsr    DoSaverJob        ;Bildschirmschoner aktivieren.

                lda    #%01000000        ;Bildschirmschoner neu starten.
                sta    Flag_ScrSaver

```

```

::52      ldx    #0                ;Register ":r0" bis ":r3"
          pla                    ;zurückschreiben.
          sta    r0,x
          inx
          cpx    #(r15H - r0L) +1
          bne    :52

          sei                    ;IRQ abschalten.
          ldx    CPU_DATA         ;CPU-Register zwischenspeichern und
          lda    #IO_IN          ;I/O-Bereich einblenden.
          sta    CPU_DATA

::53      lda    #$00
          sta    $dc00            ;Tastenregister aktivieren.
          lda    $dc01            ;Tastenstatus einlesen.
          eor    #$ff            ;Taste noch gedrückt ?
          bne    :53             ;Ja, Warteschleife...

          stx    CPU_DATA         ;CPU-Register zurücksetzen.
          plp                    ;IRQ zurücksetzen und
          rts                    ;Ende...

;*** Bildschirmschoner-Grafik.

;--- Max. Anzahl Sterne.
:MaxStars = 200

;--- Zeropage-Adressen sichern.
;(Max. 127 Bytes!)
:zpageRegAdr b $22,$23,$26,$27,$28,$29,$56,$61
             b $62,$63,$64,$65,$66,$67,$68,$69
             b $6a,$6b,$6c,$6d,$6e,$6f,$70,$8b
             b $8c,$8d,$8e,$8f

:zpageRegEnd
:zpageRegCount = (zpageRegEnd - zpageRegAdr)

:DoSaverJob ldy    #0
::save     ldx    zpageRegAdr,y   ;Register, die von RND(1)-Routine
          lda    zpage,x         ;verändert werden, auf Stack
          pha                    ;zwischenspeichern.
          iny
          cpy    #zpageRegCount
          bne    :save

          ldx    CPU_DATA
          lda    #IO_IN          ;I/O-Bereich einblenden.
          sta    CPU_DATA

          lda    $d015            ;Sprites abschalten.
          pha
          lda    $d020
          pha
          lda    #$00
          sta    $d015
          sta    $d020

```



```

        stx     CPU_DATA

        jsr     PosScreenGrafx    ;Bildschirm-Inhalt retten.
        jsr     StashRAM
        jsr     PosScreenColor
        jsr     StashRAM

        jsr     i_FillRam         ;Sternen-Farbe setzen.
        w       1000
        w       COLOR_MATRIX
        b       $10
        jsr     i_FillRam         ;Sternenhimmel löschen.
        w       8000
        w       SCREEN_BASE
        b       $00

::80      jsr     StarField

        jsr     PosScreenGrafx    ;Bildschirm-Inhalt zurücksetzen.
        jsr     FetchRAM
        jsr     PosScreenColor
        jsr     FetchRAM

        ldx     CPU_DATA         ;Sprites einschalten.
        lda     #IO_IN           ;I/O-Bereich einblenden.
        sta     CPU_DATA
        pla
        sta     $d020            ;Randfarbe VIC wiederherstellen
        pla
        sta     $d015
        stx     CPU_DATA         ;Register wieder zurücksetzen C64

::load    ldy     #zpageRegCount -1
        ldx     zpageRegAdr,y    ;Register, die von RND(1)-Routine
        pla     ;verändert werden, wieder vom Stack
        sta     zpage,x          ;einlesen und speichern.
        dey
        bpl     :load

        rts

;*** Bildschirm-Inhalt retten.
:PosScreenGrafx    LoadW            r0,SCREEN_BASE
        LoadW    r1,R2_ADDR_SS_GRAFX
        LoadW    r2,R2_SIZE_SS_GRAFX
        lda     MP3_64K_SYSTEM
        sta     r3L
        rts

:PosScreenColor    LoadW            r0,COLOR_MATRIX
        LoadW    r1,R2_ADDR_SS_COLOR
        LoadW    r2,R2_SIZE_SS_COLOR
        lda     MP3_64K_SYSTEM
        sta     r3L
        rts

```

```

;*** Tabelle mit Zufallszahlen erstellen.
;Dazu wird intern die Routine RND(1) des BASIC-Interpreters
;verwendet um Zufallszahlen im Register SEED ($008b-$008f) zu
;erstellen. Dabei werden Zufallszahlen im Bereich 0-255 erstellt,
;wobei jede Zahl nur 1x vorkommt.
:EditRandomTab lda    CPU_DATA        ;CPU-Register zwischenspeichern und
                pha                    ;BASIC-Kernal einblenden.
                lda    #KRNL_BAS_IO_IN
                sta    CPU_DATA

                lda    #%11001100      ;Startwert für RND-Funktion.
                sta    $8b
                eor    #%00110011
                sta    $8c
                eor    #%10101010
                sta    $8d
                eor    #%00011101
                sta    $8e
                eor    #%11100010
                sta    $8f

                ldy    #$00            ;Zeiger auf Tabelle löschen.

::51            tya                    ;Tabellenzeiger zwischenspeichern.
                pha

::52            jsr    $e0be          ;RND(1)-Funktion aufrufen.

                lda    $8e            ;Zufallszahl von 0-255 erstellen.
                asl
                eor    $8c
                asl
                eor    $8d
                asl
                eor    $8f

::53            ldx    #$00            ;Zeiger auf Zahlentabelle.
                cmp    RandomTab,x    ;Ist Zahl bereits in Tabelle ?
                bne    :54            ;Nein, weiter...
                lda    RandomTab,x    ;Ist Zahl = $00 ?
                bne    :52            ;Nein, neue Zahl suchen.
                beq    :55            ;Ja, Zahl speichern.
::54            inx                    ;Zeiger nächste Zahl in Tabelle.
                cpx    #MaxStars
                bne    :53            ;Weitersuchen.

::55            tax                    ;Aktuelle Zufallszahl retten.
                pla                    ;Zeiger auf Zahlentabelle wieder
                tay                    ;in yReg kopieren.
                txa                    ;Neue Zufallszahl in Zahlentabelle
                sta    RandomTab,y    ;kopieren.
                iny                    ;256 Zufallszahlen erstellt ?
                cpy    #MaxStars
                bne    :51            ;Nein, weiter...

```

```

        pla                ;CPU-Status zurücksetzen.
        sta CPU_DATA
        rts

;*** Zufallszahl aus Tabelle einlesen.
;Um die Zufallszahlen auffälliger zu
;verteilen wird die Zufallszahl mit
;dem Rasterzeilen-Register verknüpft.
:InitRandom    lda CPU_DATA    ;CPU-Register zwischenspeichern und
        pha
        lda #IO_IN            ;I/O-Bereich einblenden.
        sta CPU_DATA

        ldx r0L                ;Letzte Zufallszahl = $00 ?
        bne :51                ;Nein, weiter...
        ldx $d012              ;Rasterzeilen-Register als Zeiger.
        lda RandomTab,x        ;Neue Zufallszahl aus Tabelle

::51
holen.        eor $d012        ;Mit Rasterzeilen-Reg. verknüpfen.
        tax                    ;Als neuen Zeiger auf Tabelle ver-
        lda RandomTab,x        ;wenden und endgültige Zufallszahl
        sta r0L                ;einlesen und zwischenspeichern.

        pla                ;CPU-Status zurücksetzen.
        sta CPU_DATA
        rts

;*** Sternenfeld zeichnen.
:StarField     jsr EditRandomTab ;Zufallszahlen erstellen.
        jsr GetXYKoord          ;Startwerte für Sterne erstellen.

::51          ldy #0            ;Zeiger auf ersten Stern.
::52          tya                ;Sternzähler zwischenspeichern.
        pha
        jsr MoveStar            ;Stern zeichnen und verschieben.
        pla                    ;Sternzähler zurücksetzen.
        tay

        ldx CPU_DATA            ;CPU-Register zwischenspeichern und
        lda #IO_IN            ;I/O-Bereich einblenden.
        sta CPU_DATA

        lda $d012                ;Warteschleife.
::53          cmp $d012
        beq :53

        lda #$00
        sta $dc00                ;Tastenregister aktivieren.
        lda $dc01                ;Tastenstatus einlesen.

        stx CPU_DATA            ;CPU-Register zurücksetzen.

        eor #$ff                ;Wurde Taste gedrückt ?
        bne :54                ;Ja, Ende...

```

```

        iny                                ;Zeiger auf nächsten Stern.
        cpy    #150                        ;Alle Sterne aufgebaut ?
        bne    :52                        ;Nein, weiter...
        jmp     :51                        ;Schleife bis Taste gedrückt.
::54      rts

;*** Startwerte für alle Sterne berechnen.
:GetXyKoord    ldy    #$00
::51          jsr     SetStartKoord        ;Startwerte für aktuellen Stern.
            iny      ;Alle Sterne berechnet ?
            cpy     #MaxStars
            bne     :51                    ;Nein, weiter...
            rts

;*** Startwerte für aktuellen Stern neu setzen.
:SetStartKoord    tya
            and     #%00000111

            clc
            adc     #160 -4                ;X-Startposition von 160-167,
            sta     Star_x_l,y            ;damit nicht alle Sterne am
            lda     #0                    ;gleichen Punkt beginnen.
::80          sta     Star_x_h,y
            tya
            and     #%00000011            ;Y-Startposition von 100-103,
            clc                            ;damit nicht alle Sterne am
            adc     #100 -2                ;gleichen Punkt beginnen.
            sta     Star_y,y

            ldx     #%10001111            ;Zwangsrichtung bestimmen.
            tya
            lsr
            bcc     :51                    ;Um eine gleichmäßigere Verteilung
            ldx     #%10000011            ;der Sterne auf dem Bildschirm zu
::51          txa                            ;erreichen, wird jeder zweite Stern
            sta     r1L                    ;extrem flach, bzw. extrem steil
            eor     #%00001100            ;berechnet. Sonst erscheinen die
            sta     r1H                    ;Sterne in den Bildschirmecken
            ;konzentriert (X-Effekt).

            lda     DeltaX,y              ;Letzten Richtungswert an die
            sta     r0L                    ;Zufallszahlen-Routine übergeben.

            jsr     InitRandom              ;Neuen Richtungswert bestimmen.

            lda     r0L                    ;Sternenrichtung und
            and     r1L                    ;Geschwindigkeit eingrenzen.
            cpy     #$08                    ;Sterne #8-#15 fast vertikal.
            bcc     :52
            cpy     #$10
            bcs     :52
            ora     #%01111111
::52          sta     DeltaX,y              ;Neuen Richtungswert speichern.
            sta     DeltaXuse,y

```

```

übergeben.      lda    DeltaX,y      ;Letzten Richtungswert einlesen und
sta    r0L      ;an Zufallszahlen-Routine

                jsr    InitRandom    ;Neuen Richtungswert bestimmen.

                lda    r0L          ;Sternenrichtung und
and     r1H      ;Geschwindigkeit eingrenzen.
cpy     #$08     ;Sterne #0-#7 fast horizontal.
bcs     :53
ora     #%01111111
::53          sta    DeltaY,y      ;Neuen Richtungswert speichern.
sta     DeltaYuse,y
rts

;*** Sternen-Koordinaten einlesen.
:SetStarKoord  lda     Star_x_l,y
sta     r3L
lda     Star_x_h,y
sta     r3H
lda     Star_y ,y
sta     r11L
rts

;*** Stern verschieben.
:MoveStar     lda     DeltaXuse,y    ;Zähler für x-Richtung einlesen.
and     #%01111111                ;Neue x-Position setzen ?
beq     :51                        ;Ja, weiter...

                lda     DeltaXuse,y    ;Zähler für x-Richtung korrigieren.
and     #%10000000
sta     r0L

                lda     DeltaXuse,y
and     #%01111111
sec
sbc     #$01
ora     r0L
sta     DeltaXuse,y

                jmp     :56            ;Weiter mit y-Richtung.

::51          jsr     ClrStar        ;Aktuellen Stern löschen.

                lda     DeltaX,y      ;Zähler für x-Richtung neu
sta     DeltaXuse,y                ;initialisieren.
bmi     :52                        ; => Stern fliegt in Gegenrichtung.

                lda     Star_x_l,y    ;Stern nach rechts bewegen.
clc
adc     #$01
sta     Star_x_l,y
lda     Star_x_h,y
adc     #$00
sta     Star_x_h,y
jmp     :53

```

```

::52      lda    Star_x_l,y      ;Stern nach links bewegen.
          sec
          sbc    #$01
          sta    Star_x_l,y
          lda    Star_x_h,y
          sbc    #$00
          sta    Star_x_h,y

::53      lda    Star_x_l,y
          ora    Star_x_h,y      ;Hat Stern linken Rand erreicht ?
          beq    :55             ;Ja, neuen Stern berechnen.

          lda    Star_x_h,y
          cmp    #> 320
          bne    :54
          lda    Star_x_l,y
          cmp    #< 320
          bne    :56             ;Hat Stern rechten Rand erreicht ?
                                   ;Nein, weiter.

::54      jmp    SetStartKoord   ;Neue Sternen-Koordinate berechnen.

::55      lda    DeltaYuse,y     ;Zähler für y-Richtung einlesen.
          and    #%01111111     ;Neue y-Position setzen ?
          beq    :57             ;Ja, weiter...

          lda    DeltaYuse,y     ;Zähler für y-Richtung korrigieren.
          and    #%10000000
          sta    r0L

          lda    DeltaYuse,y
          and    #%01111111
          sec
          sbc    #$01
          ora    r0L
          sta    DeltaYuse,y
          jmp    :61

::57      jsr    ClrStar         ;Aktuellen Stern löschen.

          lda    DeltaY,y        ;Zähler für y-Richtung neu
          sta    DeltaYuse,y     ;initialisieren.
          bmi    :58             ; => Stern fliegt in Gegenrichtung.

          lda    Star_y,y        ;Stern nach unten bewegen.
          clc
          adc    #$01
          sta    Star_y,y
          jmp    :59

::58      lda    Star_y,y        ;Stern nach oben bewegen.
          sec
          sbc    #$01
          sta    Star_y,y

```

```

::59      lda    Star_y,y      ;Hat Stern oberen Rand erreicht ?
        beq     :60           ;Ja, neuen Stern berechnen.
        cmp     #200         ;Hat Stern unteren Rand erreicht ?
        bcc     :61           ;Nein, weiter...

::60      jsr     SetStartKoord ;Neue Sternen-Koordinate berechnen.

::61      jmp     DrawStar     ;Neuen Stern einzeichnen.

;*** Stern-Pixel zeichnen.
;Routine ist kompatibel zu DrawPoint ($C133). Intern wird aber eine
;FastDrawPoint-Routine zum schnelleren zeichnen verwendet.
:DrawStar    tya             ;yReg zwischenspeichern.
            pha
            jsr     SetStartKoord ;Sternen-Koordinaten einlesen.
;            lda     #$00       ;Flag für DrawPoint setzen.
;            sec         ;Flag für "Pixel setzen".
            jsr     DrawPointXL   ;Pixel zeichnen.
            pla         ;yReg zurücksetzen.
            tay
            rts

;*** Stern-Pixel löschen.
;Routine ist kompatibel zu DrawPoint ($C133). Intern wird aber eine
;FastDrawPoint-Routine zum schnelleren zeichnen verwendet.
:ClrStar     tya             ;yReg zwischenspeichern.
            pha
            jsr     SetStartKoord ;Sternen-Koordinaten einlesen.
;            lda     #$00       ;Flag für DrawPoint setzen.
;            clc         ;Flag für "Pixel löschen".
            jsr     DrawPointXL   ;Pixel zeichnen.
            pla         ;yReg zurücksetzen.
            tay
            rts

;*** Schnelle ":DrawPoint"-Routine.
:DrawPointXL  php             ;Pixel-Modus zwischenspeichern.

            lda     r11L        ;Grafikzeile #0-#24 berechnen.
            lsr
            lsr
            lsr
            tax
            lda     SCREEN_LINE_L,x
            sta     r2L
            lda     SCREEN_LINE_H,x
            sta     r2H

            lda     r3H         ;Spalte #0-#39 berechnen.
            lsr
            lda     r3L
            ror
            lsr
            lsr
            tax

```

```

        lda    SCREEN_COLUMN_L,x
        clc
        adc    r2L
        sta    r2L
        lda    SCREEN_COLUMN_H,x
        adc    r2H
        sta    r2H

        lda    r11L                ;Pixelzeile #0-#7 berechnen.
        and    #%00000111
        clc
        adc    r2L
        sta    r2L
        bcc    :51
        inc    r2H

::51      lda    r3L                ;Pixelspalte #0-#7 berechnen.
        and    #%00000111
        tax
        lda    SingleBitTab,x      ;Maske für aktuellen Pixel aus
        ldy    #$00                ;Tabelle einlesen.

        plp
        bcc    :52                ;Pixel setzen/löschen ?
        ora    (r2L),y             ; => löschen, weiter...
        sta    (r2L),y             ;Pixel setzen.
        rts

::52      eor    #$ff                ;Pixel löschen.
        and    (r2L),y
        sta    (r2L),y
        rts

;*** Zwischenspeicher
:Star_x_l    s MaxStars
:Star_x_h    s MaxStars
:Star_y      s MaxStars
:DeltaX      s MaxStars
:DeltaY      s MaxStars
:DeltaXuse   s MaxStars
:DeltaYuse   s MaxStars
:RandomTab   s MaxStars

;*** Maskentabelle zum setzen/löschen von Bits.
:SingleBitTab b $c0,$60,$30,$18,$0c,$06,$03,$03

;*** Startadressen der Grafikzeilen.
:SCREEN_LINE_L b < SCREEN_BASE + 0*8*40
                b < SCREEN_BASE + 1*8*40
                b < SCREEN_BASE + 2*8*40
                b < SCREEN_BASE + 3*8*40
                b < SCREEN_BASE + 4*8*40
                b < SCREEN_BASE + 5*8*40
                b < SCREEN_BASE + 6*8*40
                b < SCREEN_BASE + 7*8*40

```



```

b < SCREEN_BASE + 8*8*40
b < SCREEN_BASE + 9*8*40
b < SCREEN_BASE +10*8*40
b < SCREEN_BASE +11*8*40
b < SCREEN_BASE +12*8*40
b < SCREEN_BASE +13*8*40
b < SCREEN_BASE +14*8*40
b < SCREEN_BASE +15*8*40
b < SCREEN_BASE +16*8*40
b < SCREEN_BASE +17*8*40
b < SCREEN_BASE +18*8*40
b < SCREEN_BASE +19*8*40
b < SCREEN_BASE +20*8*40
b < SCREEN_BASE +21*8*40
b < SCREEN_BASE +22*8*40
b < SCREEN_BASE +23*8*40
b < SCREEN_BASE +24*8*40

:SCREEN_LINE_H b > SCREEN_BASE + 0*8*40
b > SCREEN_BASE + 1*8*40
b > SCREEN_BASE + 2*8*40
b > SCREEN_BASE + 3*8*40
b > SCREEN_BASE + 4*8*40
b > SCREEN_BASE + 5*8*40
b > SCREEN_BASE + 6*8*40
b > SCREEN_BASE + 7*8*40
b > SCREEN_BASE + 8*8*40
b > SCREEN_BASE + 9*8*40
b > SCREEN_BASE +10*8*40
b > SCREEN_BASE +11*8*40
b > SCREEN_BASE +12*8*40
b > SCREEN_BASE +13*8*40
b > SCREEN_BASE +14*8*40
b > SCREEN_BASE +15*8*40
b > SCREEN_BASE +16*8*40
b > SCREEN_BASE +17*8*40
b > SCREEN_BASE +18*8*40
b > SCREEN_BASE +19*8*40
b > SCREEN_BASE +20*8*40
b > SCREEN_BASE +21*8*40
b > SCREEN_BASE +22*8*40
b > SCREEN_BASE +23*8*40
b > SCREEN_BASE +24*8*40

;*** Startadressen der Grafikspalten.
:SCREEN_COLUMN_L b < 8 * 0
b < 8 * 1
b < 8 * 2
b < 8 * 3
b < 8 * 4
b < 8 * 5
b < 8 * 6
b < 8 * 7
b < 8 * 8
b < 8 * 9

```

```
b < 8 * 10
b < 8 * 11
b < 8 * 12
b < 8 * 13
b < 8 * 14
b < 8 * 15
b < 8 * 16
b < 8 * 17
b < 8 * 18
b < 8 * 19
b < 8 * 20
b < 8 * 21
b < 8 * 22
b < 8 * 23
b < 8 * 24
b < 8 * 25
b < 8 * 26
b < 8 * 27
b < 8 * 28
b < 8 * 29
b < 8 * 30
b < 8 * 31
b < 8 * 32
b < 8 * 33
b < 8 * 34
b < 8 * 35
b < 8 * 36
b < 8 * 37
b < 8 * 38
b < 8 * 39
```

```
:SCREEN_COLUMN_H      b > 8 * 0
```

```
b > 8 * 1
b > 8 * 2
b > 8 * 3
b > 8 * 4
b > 8 * 5
b > 8 * 6
b > 8 * 7
b > 8 * 8
b > 8 * 9
b > 8 * 10
b > 8 * 11
b > 8 * 12
b > 8 * 13
b > 8 * 14
b > 8 * 15
b > 8 * 16
b > 8 * 17
b > 8 * 18
b > 8 * 19
b > 8 * 20
b > 8 * 21
b > 8 * 22
b > 8 * 23
```

```
b > 8 * 24
b > 8 * 25
b > 8 * 26
b > 8 * 27
b > 8 * 28
b > 8 * 29
b > 8 * 30
b > 8 * 31
b > 8 * 32
b > 8 * 33
b > 8 * 34
b > 8 * 35
b > 8 * 36
b > 8 * 37
b > 8 * 38
b > 8 * 39
```

```
,*****
/
*** Endadresse testen.
/
*****
g LD_ADDR_SCRSAVER + R2_SIZE_SCRSAVER -1
*****
/
```

M.3 GEOS/MegaPatch: "geoPainterViewer"

Wie im **Teil C, Anhang G.3 ab Seite 412** erwähnt, verfügt GEOS über keinerlei Routinen um eine GeoPaint-Datei auf dem Bildschirm anzuzeigen. Damit Anwender von GEOS/MegaPatch64 das Rad nicht neu erfinden müssen, hier ein Beispiel für einen GeoPaint-Viewer. Die Routine ist für GEOS64 ausgelegt, für GEOS64 und GEOS128 findet sich eine entsprechende Routine im Quelltext zum GEOS.Editor (siehe Datei s.MP3.Edit.1 ab dem Label »:ViewPaintFile«).

Unter GEOS/MegaPatch128 wäre es denkbar, die Datei auf dem 40Z-Bildschirm anzuzeigen und dann in den VDC zu kopieren.

```
;
; Symboltabellen einbinden.
;
if .p
    t "TopSym"
    t "TopSym.MP3"
    t "TopSym.IO"
    t "TopMac"
endif

;
; GEOS-Header definieren.
;
    n "geoPainterViewer"
    c "PainterViewer V1.0",NULL
    a "Markus Kanet",NULL

    f APPLICATION
    z $80 ;Nur GEOS64.

    o APP_RAM
    p MAININIT

    h "GeoPainter-Viewer für GEOS/MegaPatch64"

    i
    
;
; GeoPainter-Viewer einbinden.
;
    t "inc.ReadGPFile"
;

;
; Hauptprogramm.
;
:MAININIT    jsr    GetBackScreen    ;Hintergrund initialisieren.
```

```

::doDlgBox      LoadW  r0,dlgSlctFile
                LoadW  r5,dataFileName      ;Ablagebereich Dateiname.
                LoadB  r7L,APPL_DATA        ;GEOS-Filetyp: APPL_DATA/Dokument.
                LoadW  r10,PaintClass       ;GEOS-Klasse : "Paint Image "
                jsr     DoDlgBox             ;Dateiauswahlbox anzeigen.

                lda     sysDBData           ;Laufwerk wechseln?
                bpl     :nodrive             ; => Nein, weiter...
                and     #%00001111
                jsr     SetDevice           ;Laufwerk aktivieren.
                txa
                bne     :exit               ; Fehler?
                beq     :doDlgBox           ; => Ja, Abbruch...
                ;Dialogbox erneut anzeigen.

::nodrive      cmp     #CANCEL             ;Abbruch gewählt?
                beq     :exit               ; => Ja, Ende...

                php
                sei                         ;Interrupt sperren.

                ldx     CPU_DATA            ;I/O-Bereich einblenden.
                lda     #IO_IN
                sta     CPU_DATA

::nokey        lda     #$00                ;Warten bis keine Taste gedrückt.
                sta     $dc00
                lda     $dc01
                eor     #$ff
                bne     :nokey

                stx     CPU_DATA            ;I/O-Bereich ausblenden.

                plp                         ;Interrupt-Status zurücksetzen.

                LoadB  a0L,$80             ;Farben löschen, $00=Nicht löschen.
                LoadW  a2 ,buffer          ;1448-Byte-Zwischenspeicher.
                jsr     ViewPaintFile       ;GeoPaint-Datei anzeigen.

                php
                sei                         ;Interrupt sperren.

                ldx     CPU_DATA            ;I/O-Bereich einblenden.
                lda     #IO_IN
                sta     CPU_DATA

::wait         lda     #$00                ;Warten auf Tastendruck.
                sta     $dc00
                lda     $dc01
                eor     #$ff
                beq     :wait
                stx     CPU_DATA            ;I/O-Bereich ausblenden.

                plp                         ;Interrupt-Status zurücksetzen.

::exit         jmp     MAININIT            ;Nächste Datei anzeigen.
                jmp     EnterDesktop        ;Zurück zum Desktop.

```

```

;
; Dateiauswahlbox.
;
;:dlgSlctFile    b $81
;                b DBGETFILES ! DBSETDRVICON
;                b  $00,$00
;
;                b OPEN
;                b  $00,$00
;                b CANCEL
;                b  $00,$00
;
;                b NULL
;
;
; GEOS-Klasse GeoPaint-Dokumente.
;
;:PaintClass     b "Paint Image ",NULL
;
;
; Zwischenspeicher für Grafikdaten.
;
;:buffer

```

Das Hauptprogramm wählt nur die GeoPaint-Datei zur Anzeige aus. Der eigentliche GeoPaint-Viewer wird im Quelltext über den Opcode *t "src.ReadGPFile"* als Include-File in den Quelltext eingebunden.

Hier der Quelltext der eigentlichen GeoPaint-Viewer-Routine:

```

;*** Hintergrundbild anzeigen.
;
;Übergabe:  dataFileName = Name GeoPaint-Datei.
;          a0L = $00: Farb-RAM nicht löschen.
;          $80: Farb-RAM löschen.
;          a2  = Puffer für GeoPaint-Daten (2*80*8+8+2*80 = 1448 Bytes)
;
;Verwendet: a2  = Zeiger auf Grafikdaten Zeile #1.
;          a3  = Zeiger auf Grafikdaten Zeile #2.
;          a4  = Zeiger auf 8Byte-Datenspeicher.
;          a5  = Zeiger auf Farbdaten Zeile #1.
;          a6  = Zeiger auf Farbdaten Zeile #2.
;
;:ViewPaintFile ldx    #0                ;Zeiger berechnen für:
;:1            lda    a2L                ; - Grafikzeile #1
;            clc                    ; - Grafikzeile #2
;            adc    scrnBaseData +0,x ; - Farbzeile #1
;            sta    a3L,x              ; - Farbzeile #2
;            lda    a2H
;            adc    scrnBaseData +1,x
;            sta    a3H,x
;            inx
;            inx
;            cpx    #8
;            bcc    :1

```

```

        bit    a0L                ;Farb-RAM löschen?
        bpl    :load              ; => Nein, weiter...
        jsr    GetBorderCol

::load      LoadW  r0,dataFileName
        jsr    OpenRecordFile    ;geoPaint-Dokument öffnen.
        txa                    ;Fehler?
        bne    :53                ; => Ja, Abbruch...

        bit    a0L                ;Farb-RAM löschen?
        bpl    :50                ; => Nein, weiter...

        lda    backScrnCol
        jsr    i_UserColor        ;Farb-RAM löschen.
        b      $00,$00,$28,$19

::50        LoadW  r14,SCREEN_BASE ;Zeiger auf Grafikspeicher.
        LoadW  r15,COLOR_MATRIX

        lda    #$00
::51        sta    a9H            ;VLIR-Datensatz-Nr.

        jsr    Get80Cards         ;Grafikzeile einlesen.
        jsr    Prnt_Grfx_Cols    ;Grafikzeile ausgeben.

        inc    a9H                ;Nächster Datensatz.
        lda    a9H
        cmp    usedRecords        ;Ende geoPaint-Dokument erreicht?
        bcs    :52                ; => Ja, Ende...
        cmp    #13                ;Bildschirm voll?
        bcc    :51                ; => Nein, weiter...

::52        ldx    #NO_ERROR
::53        txa
        pha
        jsr    CloseRecordFile    ;geoPaint-Dokument schließen.
        pla
        tax
        rts

;
; Startadresse Daten in VLIR-Datensatz.
;
::scrnBaseData w 640
               w 640 +640
               w 640 +640 +8
               w 640 +640 +8 +80

;*** Rahmenfarbe einlesen.
::GetBorderCol php                ;Hintergrundfarbe löschen.
               sei

               ldx    CPU_DATA
               lda    #$35          ;I/O-Bereich aktivieren.
               sta    CPU_DATA

```

```

        lda    extclr          ;Rahmenfarbe einlesen.
        and    #%00001111     ;Rahmenfarbe isolieren.
        sta    r0L

        asl                      ;Farbe für Vorder- und
        asl                      ;Hintergrundfarbe berechnen.
        asl
        ora    r0L
        sta    backScrnCol

        stx    CPU_DATA        ;I/O-Bereich ausblenden.

        plp
        rts

;
; Zwischenspeicher Hintergrundfarbe
;
:backScrnCol  b $00

;*** Grafikdaten ausgeben.
;Eine geoPaint-Zeile besteht aus zwei
;Grafikzeilen a 8 Pixel Höhe.
:Prnt_GrFx_Cols      lda      a2L          ;Zeile #1 ausgeben.
                    ldx      a2H
                    jsr      MoveGrfx
                    lda      a5L
                    ldx      a5H
                    jsr      MoveCols

                    lda      a9H          ;12*2 +1 Zeilen.
                    cmp      #12
                    bcs      :1

                    lda      a3L          ;Zeile #2 ausgeben.
                    ldx      a3H
                    jsr      MoveGrfx
                    lda      a6L
                    ldx      a6H
                    jsr      MoveCols

::1                rts

;*** Grafikdaten in Bildschirm kopieren.
:MoveGrfx          sta    r0L          ;Zeiger auf C64-Grafikspeicher.
                    stx    r0H

                    LoadW    r2,40*8    ;Anzahl Bytes in einer Zeile.

                    lda    r14L          ;Startadresse Zwischenspeicher
                    sta    r1L          ;setzen und Position für nächste
                    clc                  ;Grafikzeile berechnen.
                    adc    r2L
                    sta    r14L

```



```

        lda    r14H
        sta    r1H
        adc    r2H
        sta    r14H

        jmp    MoveData          ;Grafikdaten kopieren.

;*** Farbdaten in Bildschirm kopieren.
:MoveCols    sta    r0L          ;Zeiger auf C64-Farbspeicher.
             stx    r0H

             LoadW   r2,40      ;Anzahl Bytes in einer Zeile.

             lda    r15L        ;Startadresse Zwischenspeicher
             sta    r1L        ;setzen und Position für nächste
             clc                ;Grafikzeile berechnen.
             adc    r2L
             sta    r15L
             lda    r15H
             sta    r1H
             adc    r2H
             sta    r15H

             jmp    MoveData          ;Farbdaten kopieren.

;*** Eine Grafikzeile (80 Cards/8 Pixel hoch) einlesen.
:Get80Cards   jsr    PointRecord  ;Zeiger auf Grafikzeile.
             txa                ;Fehler?
             bne    NoGrfxData    ; => Ja, Abbruch...
             tya
             bne    LoadVLIR_Data

:NoGrfxData   LoadW   r0,1448    ;Leere Grafikzeile ausgeben.
             MoveW   a2,r1
             LoadB   r2L,NULL
             jmp     FillRam

;*** Grafikbytes aus Datensatz einlesen.
:LoadVLIR_Data LoadW   r4,diskBlkBuf ;Zeiger auf Diskettenspeicher.
             jsr    GetBlock      ;Ersten Sektor des aktuellen
             txa                ;Datensatzes einlesen. Fehler ?
             bne    NoGrfxData    ; => Ja, nächste Zeile...

             MoveW   a2,r0        ;Zeiger auf Grafikdatenspeicher.

             ldx    #$01          ;Zeiger auf erstes Byte in Datei.
             stx    r5H

:GetNxDataByte jsr    GetNxByte    ;Nächstes Byte einlesen.
             sta    r2H          ;Byte zwischenspeichern.

             ldy    #$00
             bit    r2H          ;Gepackte Daten ?
             bmi    GetPackedBytes ;Ja, weiter...

```

```

        lda    r2H
        and    #$3f                ;Anzahl Bytes ermitteln.
        beq    EndOfData           ;$00 = Keine Daten.
        sta    r2H                 ;Anzahl Bytes merken.
        bvs    Repeat8Byte         ;Bit #6 = 1, 8-Byte-Packformat.

::1      jsr    GetNxByte           ;Byte einlesen und in Grafikdaten-
        sta    (r0L),y             ;speicher kopieren.
        iny
        cpy    r2H                 ;Alle Bytes gelesen ?
        bne    :1                  ;Nein, weiter...

;*** Zeiger auf Grafikdatenspeicher korrigieren.
:SetNewMemPos  tya                ;Zeiger auf Grafikdatenspeicher
                clc                ;korrigieren.
                adc    r0L
                sta    r0L
                bcc    GetNxDataByte
                inc    r0H
                bne    GetNxDataByte ;Nächstes Byte einlesen.

:EndOfData    rts

;*** 8-Byte-Daten wiederholen.
:Repeat8Byte  jsr    GetNxByte     ;Nächstes Byte aus Datensatz
                sta    (a4L),y     ;einlesen und in Zwischenspeicher.
                iny                ;Zeiger auf nächstes Byte.
                cpy    #$08        ;8 Byte eingelesen ?
                bne    Repeat8Byte ;Nein, weiter...

::1          ldx    #$00
::2          ldy    #$07           ;8 Byte in Grafikdatenspeicher.
                lda    (a4L),y
                sta    (r0L),y
                dey
                bpl    :2
                lda    r0L         ;Zeiger auf Grafikdatenspeicher
                                ;korrigieren.
                clc
                adc    #$08
                sta    r0L
                bcc    :3
                inc    r0H
::3          inx                ;Anzahl Wiederholungen +1.
                cpx    r2H         ;Wiederholungen beendet ?
                bne    :1          ;Nein, weiter...
                beq    GetNxDataByte ;Weiter mit nächstem Byte.

;*** Gepackte Daten einlesen.
:GetPackedBytes

                lda    r2H         ;Anzahl gepackte Daten berechnen.
                and    #$7f
                beq    EndOfData    ;$00 = Keine Daten, Ende...
                sta    r2H         ;Anzahl Bytes merken.
                jsr    GetNxByte    ;Datenbyte einlesen.

```

```

        ldy    r2H
::1      dey    (r0L),y      ;Byte in Grafikdatenspeicher
        dey    ;kopieren (Anzahl in ":r2H")
        bpl    :1

        ldy    r2H          ;Zeiger auf Grafikdatenspeicher
        bne    SetNewMemPos ;korrigieren.

;*** Nächstes Byte aus Paint-Datei einlesen.
:GetNxByte    ldx    r5H
              inx
              bne    RdBytFromSek
              lda    r1L
              bne    GetNxSektor

:GfxLoadError jmp    NoGrfxData      ;Leere Zeile ausgeben.

;*** Nächsten Sektor aus Paint-Datensatz einlesen.
:GetNxSektor  sty    a9L

              lda    diskBlkBuf +0    ;Zeiger auf nächsten Sektor.
              sta    r1L
              lda    diskBlkBuf +1
              sta    r1H
              jsr    GetBlock          ;Sektor einlesen.
              txa                      ;Diskettenfehler ?
              bne    GfxLoadError      ; => Ja, Abbruch...

              ldy    a9L
              ldx    #$02              ;Zeiger auf erstes Byte in Sektor.

;*** Nächstes Byte aus Sektor einlesen.
:RdBytFromSek lda    r1L              ;Letzter Sektor?
              bne    :1                ; => Nein, weiter....
              cpx    r1H              ;Letztes Bytes aus letztem Sektor?
              bcc    :1                ; => Nein, weiter....
              bne    GfxLoadError      ; => Ja, Abbruch....

::1        lda    diskBlkBuf,x        ;Byte aus Sektor einlesen.
              stx    r5H              ;Bytezeiger speichern.
              rts

```

M.4 Demo/DeskAccessory: "geoScreenCapture"

Das Demo "geoPainter" ist eine Application und zeigt GeoPaint-Dokumente am Bildschirm an, verwendet also Routinen zur Anzeige von GeoPaint-Dokumenten.

Um GeoPaint-Dokumente auch aus einer Anwendung heraus erstellen zu können, wurde das DeskAccessory "geoScreenCapture" entwickelt. Auch dieses Programm ist lediglich als Demo zu verstehen.

Das Programm beinhaltet Routinen zum schreiben von GeoPaint-Dokumenten, die in ähnlicher Form auch in GEOS/MegaPatch im TaskManager enthalten sind. Damit lassen sich unter GEOS Screenshots im GeoPaint-Format erstellen.

Das Programm kann außerdem auch ein Photoscrap aus einem Teil des angezeigten GEOS-Bildschirms erstellen, es ist sogar möglich den ganzen Bildschirm als Photoscrap zu speichern.

Da es sich hier lediglich um eine Demo-Anwendung handelt, um die Routinen zum erstellen von GeoPaint-Dokumenten bzw. Photoscraps zu demonstrieren, wurde das Programm relativ einfach gehalten. Es funktioniert nur unter GEOS64.

Nach dem Start erscheint ein kleines Rechteck, das den aktuellen Ausschnitt für das Photoscrap darstellt. Über die Cursor-Tasten kann der Ausschnitt in 8-Pixel-Schritten verschoben werden. Die 8er-Schritte sind erforderlich, da auch Farbinformationen im Photoscrap gespeichert werden sollen.

Über die Tasten [X], [SHIFT]+[X], [Y] und [SHIFT]+[Y] lässt sich die Größe des Ausschnitts ändern. Über die Tasten [M] und [SHIFT]+[M] kann entweder die max. Größe eines Photoscrap für GeoPaint/GeoWrite gesetzt oder der ganze Bildschirm für das Photoscrap ausgewählt werden.

Über die Taste [RETURN] wird der gewählte Bildausschnitt in ein Photoscrap gespeichert und das Programm anschließend beendet.

Über die Taste [C] lässt sich der GEOS-Bildschirm inkl. Farbinformationen in ein GeoPaint-Dokument speichern.

Beim schreiben des Programms sind dann noch zusätzliche Informationen in dieses Handbuch eingeflossen, unter anderen auch zum Format der GeoPaint-Dokumente und zu deren Infoblock.

Das Programm ist in drei Teile aufgeteilt: Das Hauptprogramm, die Routinen zum erstellen eines Photoscrap und die Routinen zum schreiben von GeoPaint-Dateien.

Hier nun zuerst die Hauptanwendung. Diese übernimmt das Tastenmenü, den Ausschnitt für das Photoscrap festlegen, sowie den Aufruf der eigentlichen Routinen zum schreiben der Bilddateien.

```
;
; Symboltabellen einbinden.
;
if .p
    t "TopSym"
    t "TopMac"
endif
```

```

;
; GEOS-Header .
;
        n "geoScreenCapture"
        c "Capture      V1.0"
        f DESK_ACC
        a "Markus Kanet"
        z $80 ;Nur GEOS64.

        o APP_RAM
        q END_DESC_ACC
        p MAININIT

        h "Screenshot mit c, Photoscrap mit RETURN."
        h "x/X, y/Y, m/M für Größe, Position mit CRSR-Tasten."

        i
        
;
; Quelltext für Photoscrap und
; Screenshot einbinden.
;
        t "inc.WritePScrap"      ;Photoscrap erstellen.
        t "inc.WriteGPFile"     ;Screenshot erstellen.

;
; geoScreenCapture
;
; Bildschirm als PhotoScrap oder als
; GeoPaint-Datei speichern.
;
; DeskAccessories dürfen die Register
; a0-a9 nicht verändern, daher die
; Register zwischenspeichern.
;
:MAININIT      ldx      #0          ;Register a0-a1
::l1           lda      a0,x        ;zwischenspeichern.
               sta      aBuf +0,x
               inx
               cpx      #4
               bcc      :l1

               ldx      #0          ;Register a2-a9
::l2           lda      a2,x        ;zwischenspeichern.
               sta      aBuf +4,x
               inx
               cpx      #8 *2
               bcc      :l2

               jsr      OpenDisk    ;Diskette öffnen.
               txa          ;Diskettenfehler?
               bne      MainExit    ; => Ja, Abbruch...

```

```

        jsr    defScrapSize      ;Rahmen für PhotoScrap berechnen.
        jsr    prntScrapSize     ;Rahmen um PhotoScrap zeichnen.

        lda    #< setScrapSize   ;Tastatur-Menü installieren.
        sta    keyVector +0
        lda    #> setScrapSize
        sta    keyVector +1

; Tastatur-Menü ausführen.
        rts                      ;Zurück zur Mainloop.

;
; DeskAccessory beenden. Dazu die zuvor
; gesicherten Register a0 bis a9 wieder
; zurückschreiben.
;
MainExit    ldx    #0            ;Register a0-a1
::l1        lda    aBuf +0,x      ;wieder zurückschreiben.
            sta    a0,x
            inx
            cpx    #4
            bcc    :l1

            ldx    #0            ;Register a2-a9
::l2        lda    aBuf +4,x      ;wieder zurückschreiben.
            sta    a2,x
            inx
            cpx    #8 *2
            bcc    :l2

        jsr    OpenDisk          ;Diskette öffnen.

        lda    #< RstrAppl       ;DeskAccessory über die
        sta    appMain +0        ;Mainloop beenden.
        lda    #> RstrAppl
        sta    appMain +1

        rts

;
; Größe Photoscrap anzeigen.
;
; Dabei wird am Bildschirm ein 1-Pixel
; breiter Rahmen invertiert um die
; aktuelle Größe anzuzeigen.
;
; Übergabe:
; r2L/r2H = y-Koordinate oben/unten
; r3 /r4 = x-Koordinate links/rechts
;
prntScrapSize lda    #ST_WR_FORE ;Nur in den Vordergrund zeichnen.
              sta    dispBufferOn

```

```

lda r2L ;y-Koordinaten zwischenspeichern.
pha
lda r2H
pha

lda r2L
sta r2H
jsr InvertRectangle ;Oberen Rand invertieren.

pla
sta r2H ;y-Koordinate wieder zurücksetzen.
sta r2L
jsr InvertRectangle ;Unteren Rand invertieren.
pla
sta r2L ;y-Koordinate wieder zurücksetzen.

lda r3H ;x-Koordinaten zwischenspeichern.
pha
lda r3L
pha
lda r4H
pha
lda r4L
pha

lda r3L
sta r4L
lda r3H
sta r4H
jsr InvertRectangle ;Linken Rand invertieren.

pla
sta r4L ;x-Koordinate wieder zurücksetzen.
sta r3L
pla
sta r4H
sta r3H
jsr InvertRectangle ;Rechten Rand invertieren.

pla
sta r3L
pla
sta r3H ;x-Koordinate wieder zurücksetzen.

rts

```

```

;
; X-/y-Koordinaten für den 1-Pixel
; Rahmen des Photoscrap berechnen.
;
;
; Rückgabe:
; r2L/r2H = y-Koordinate oben/unten
; r3 /r4 = x-Koordinate links/rechts
;

```

```

:defScrapSize  lda    scrapXPos      ;x-Koordinate in Cards einlesen.
               sta    r3L
               ldx    #$00
               stx    r3H

;               lda    scrapXPos      ;Breite des Photoscrap einlesen.
               clc
               adc    scrapWidth
               sta    r4L
;               ldx    #$00
               stx    r4H

               ldx    #r3L          ;Linker Rand nach Pixel
               ldy    #3            ;konvertieren.
               jsr    DShiftLeft

               ldx    #r4L          ;Rechter Rand nach Pixel
               ldy    #3            ;konvertieren.
               jsr    DShiftLeft
               ldx    #r4L          ;Rechten Rand auf das letzte Pixel
               jsr    Ddec          ;im letzten Card setzen.

               lda    scrapYPos      ;y-Koordinate einlesen.
               sta    r2L

;               lda    scrapYPos      ;Unteren Rand des Photoscrap
               clc                  ;berechnen.
               adc    scrapHeight

               sec                  ;Unteren Rand auf das letzte Pixel
               sbc    #1            ;im letzten Card setzen.
               sta    r2H
               rts

;
; Tastatur-Menü
;
; Die Routine wird über die Mainloop
; aufgerufen und werten einen Tasten-
; druck aus und ruft dann die dazu
; passende Menü-Routine auf.
;
;setScrapSize  jsr    defScrapSize    ;Größe Photoscrap-Rahmen berechnen.
               jsr    prntScrapSize  ;Photoscrap-Rahmen löschen.

               ldx    #0
               lda    keyData        ;Aktuelle Taste einlesen.
::1            cmp    keyDataTab,x   ;Taste in Tabelle suchen?
               beq    :2            ; => Gefunden, weiter...
               inx
               cpx    #MAX_KEYS      ;Alle Tasten durchsucht?
               bcc    :1            ; => Nein, weiter...

;               jsr    defScrapSize    ;Größe Photoscrap-Rahmen berechnen.
               jsr    prntScrapSize  ;Photoscrap-Rahmen anzeigen.
               rts

```



```

;
; Tasten-Routine ausführen.
;
::2      lda    adrDataTabH,x    ;Startadresse auf Stack schieben
        pha                    ;und Tasten-Routine aufrufen.
        lda    adrDataTabL,x
        pha
        rts

;
; Liste der Menütasten.
;
keyDataTab    b $0d ;RETURN
              b $63 ;c /Capture
              b $08 ;Cursor links
              b $1e ;Cursor rechts
              b $10 ;Cursor hoch
              b $11 ;Cursor runter
              b $78 ;x-kleiner
              b $58 ;x-größer
              b $79 ;y-kleiner
              b $59 ;y-größer
              b $6d ;m /GeoPaint
              b $4d ;M /Maximum
: endDataTab

;
; Anzahl der Menütasten ermitteln.
;
: MAX_KEYS      = endDataTab - keyDataTab

;
; Startadressen der Tasten-Routinen.
; Da die Adresse auf den Stack gelegt
; und als Rücksprungadresse genutzt
; wird, muss der Wert für den Stack
; umd 1 Byte reduziert werden.
;
; Lowbyte:
: adrDataTabL    b < DoPhotoScrap -1
                b < DoScreenShot -1
                b < moveLeft -1
                b < moveRight -1
                b < moveUp -1
                b < moveDown -1
                b < sizeXsub -1
                b < sizeXadd -1
                b < sizeYsub -1
                b < sizeYadd -1
                b < sizePaint -1
                b < sizeMax -1

```

```

; Highbyte:
; adrDataTabH      b > DoPhotoScrap -1
                   b > DoScreenShot -1
                   b > moveLeft -1
                   b > moveRight -1
                   b > moveUp -1
                   b > moveDown -1
                   b > sizeXsub -1
                   b > sizeXadd -1
                   b > sizeYsub -1
                   b > sizeYadd -1
                   b > sizePaint -1
                   b > sizeMax -1

;
; Screenshot erstellen
;
; Über die Taste `c` wird der aktuelle
; Bildinhalt mit Grafik+Farbe in ein
; GeoPaint-Dokument gespeichert.
;
; DoScreenShot      jsr      CREATE_GIMAGE      ;Screenshot erstellen.
                   jmp      MainExit            ;DeskAccessory beenden.

;
; Photoscrap erstellen
;
; Über `RETURN` wird die aktuelle
; Auswahl mit Grafik+Farbe in eine
; Photoscrap-Datei gespeichert.
;
; DoPhotoScrap      jsr      CREATE_PSCRAP      ;Photoscrap erstellen.
                   jmp      MainExit            ;DeskAccessory beenden.

;
; Taste `CRSR-LEFT`:
; Auswahl nach links schieben.
;
; moveLeft          ldx      scrapXPos           ;Auswahl bereits am rechten Rand?
                   beq      :done                ; => Ja, Ende...
                   dex                     ;Rahmen nach links schieben.
                   stx      scrapXPos
                   jsr      defScrapSize          ;Größe Photoscrap-Rahmen berechnen.
::done              jmp      prntScrapSize        ;Photoscrap-Rahmen anzeigen.

;
; Taste `CRSR-RIGHT`:
; Auswahl nach rechts schieben.
;
; moveRight          lda      scrapXPos           ;Auswahl bereits am rechten Rand?
                   clc
                   adc      scrapWidth
                   cmp      #40
                   bcs      :done                ; => Ja, Ende...

```

```

        inc    scrapXPos      ;Auswahl nach rechts schieben.
        jsr    defScrapSize   ;Größe Photoscrap-Rahmen berechnen.
::done    jmp    prntScrapSize ;Photoscrap-Rahmen anzeigen.

;
; Taste `CRSR-UP`:
; Auswahl nach oben schieben.
;
:moveUp    lda    scrapYPos      ;Auswahl bereits am oberen Rand?
          beq     :done          ; => Ja, Ende...
          sec
          sbc     #8             ;Auswahl nach oben schieben.
          sta    scrapYPos
          jsr    defScrapSize   ;Größe Photoscrap-Rahmen berechnen.
::done    jmp    prntScrapSize ;Photoscrap-Rahmen anzeigen.

;
; Taste `CRSR-DOWN`:
; Auswahl nach unten schieben.
;
:moveDown   lda    scrapYPos      ;Auswahl bereits am unteren Rand?
          clc
          adc     scrapHeight
          bcs     :done          ;Überlauf, Ende...
          cmp     #200
          bcs     :done          ; => Ja, Ende...

          lda    scrapYPos      ;Auswahl nach unten schieben.
          clc
          adc     #8
          sta    scrapYPos
          jsr    defScrapSize   ;Größe Photoscrap-Rahmen berechnen.
::done    jmp    prntScrapSize ;Photoscrap-Rahmen anzeigen.

;
; Taste `x`:
; Breite der Auswahl reduzieren.
;
:sizeXsub    ldx    scrapWidth    ;Breite bereit auf Minimum?
          dex
          beq     :done          ; => Ja, Ende...
          stx    scrapWidth      ;Breite der Auswahl reduzieren.
          jsr    defScrapSize   ;Größe Photoscrap-Rahmen berechnen.
::done    jmp    prntScrapSize ;Photoscrap-Rahmen anzeigen.

;
; Taste `SHIFT x`:
; Breite der Auswahl vergrößern.
;
:sizeXadd    lda    scrapXPos      ;x-Koordinate und Breite bereits
          clc                    ;am rechten Rand?
          adc     scrapWidth
          cmp     #40
          bcs     :done          ; => Ja, Ende...

```

```

        inc    scrapWidth      ;Breite der Auswahl vergrößern.
        jsr    defScrapSize    ;Größe Photoscrap-Rahmen berechnen.
::done   jmp    prntScrapSize   ;Photoscrap-Rahmen anzeigen.

;
; Taste `y`:
; Höhe der Auswahl reduzieren.
;
::sizeYsub    lda    scrapHeight    ;Höhe bereits auf Minimum?
              sec
              sbc    #8
              bcc    :done          ;Unterlauf, Ende...
              beq    :done          ; => Ja, Ende...

              sta    scrapHeight    ;Höhe der Auswahl reduzieren.
              jsr    defScrapSize    ;Größe Photoscrap-Rahmen berechnen.
::done   jmp    prntScrapSize   ;Photoscrap-Rahmen anzeigen.

;
; Taste `SHIFT y`:
; Höhe der Auswahl vergrößern.
;
::sizeYadd    lda    scrapYPos      ;y-Koordinate und Höhe bereits
              clc                    ;am unteren Rand?
              adc    scrapHeight
              bcs    :done          ;Überlauf, Ende...
              cmp    #200
              bcs    :done          ; => Ja, Ende...

              lda    scrapHeight    ;Höhe der Auswahl vergrößern.
              clc
              adc    #8
              sta    scrapHeight
              jsr    defScrapSize    ;Größe Photoscrap-Rahmen berechnen.
::done   jmp    prntScrapSize   ;Photoscrap-Rahmen anzeigen.

;
; Taste `m`:
; Größe der Auswahl auf die maximale
; Auswahl-Größe von GeoPaint setzen.
; Dieser Ausschnitt kann von GeoPaint
; und Geowrite unskaliert in das
; Dokument eingefügt werden.
;
::sizePaint    lda    #0            ;Auswahl auf GeoPaint-Größe
              sta    scrapXPos      ;setzen.
              sta    scrapYPos
              lda    #33            ;Max. $21 Cards breit.
              sta    scrapWidth
              lda    #144           ;Max. $90 Cards hoch.
              sta    scrapHeight

              jsr    defScrapSize    ;Größe Photoscrap-Rahmen berechnen.
              jmp    prntScrapSize   ;Photoscrap-Rahmen anzeigen.

```

```

;
; Taste `SHIFT m`:
; Größe der Auswahl auf den gesamten
; Bildschirm setzen.
; Das Photoscrap kann von GeoPaint dann
; nur noch skaliert eingefügt werden,
; dabei werden die Farben aber nicht
; in das Bild übernommen.
; GeoWrite kann das Photoscrap nicht
; mehr in ein Dokument einfügen.
;
:sizeMax      lda    #0          ;Auswahl auf Bildschirmgröße
              sta    scrapXPos   ;setzen.
              sta    scrapYPos
              lda    #40         ;Max. $28 Cards breit.
              sta    scrapWidth
              lda    #200        ;Max. $19 Cards hoch.
              sta    scrapHeight

              jsr    defScrapSize ;Größe Photoscrap-Rahmen berechnen.
              jmp    prntScrapSize ;Photoscrap-Rahmen anzeigen.

;*** Ende DeskAccessory / Beginn Daten.

;
; Zwischenspeicher für die Application-
; Register. Werden am Ende wieder in
; die Register a0-a9 zurückgeschrieben.
;
:aBuf          s    10 *2

; Speicher für Photoscrap/Screenshot.
:dataBuf

; Speicher für Original-Daten.
:dataUnpacked = dataBuf

; Speicher für gepackte Daten.
:dataPacked   = dataUnpacked + 1280 + 8 + 160 +1

; Größe Zwischenspeicher berechnen.
:dataBufEnd   = dataPacked   + 1280 + 8 + 160 +1 +48
:dataBufSize  = (dataBufEnd -dataBuf)

;
; Die Größe des DeskAccessory wird so
; gewählt, das auch der Datenspeicher
; im Swapfile ausgelagert wird.
;
:END_DESC_ACC = dataBufEnd

```

Am Ende des Hauptprogramms wird ein Zwischenspeicher definiert, der von den folgenden Routinen zur Ablage von Daten benötigt wird.

Es folgen die Routinen zum erzeugen eines Photoscrap:

```
;
; Photoscrap-Datei erstellen.
;
;
; Dazu wird ein 40Z-Bildausschnitt mit
; Grafik+Farbe in eine PhotoScrap-Datei
; gespeichert.
;
; Übergabe:
:scrapXPos      b $01 ;x-Position in Cards.
:scrapYPos      b $08 ;y-Position, nur ganze 8er-Blöcke!
:scrapWidth     b $03 ;Breite: In Cards!
:scrapHeight    b $18 ;Höhe : Nur ganze 8er-Blöcke!
;
; Interne Variablen:
:dirEntryBlk    b $00,$00
:dirEntryAdr    w $0000
:curBlock       b $00,$00
:curByte        b $00
:nextFreeBlk    b $00,$00
;
; Benötigter Datenspeicher:
; :dataUnpacked = max. 40x8 Byte ungepackte Daten.
; :dataPacked   = max. 40x8 Byte + ca.30 Kompressionsbyte, wenn alle
;               Datenbyte verschieden sind und ein packen unmöglich ist.
;
;
; PhotoScrap erstellen.
;
; Hauptroutine:
; - Leere Photoscrap-Datei erstellen
; - Grafikdaten packen und anhängen
; - Farbdaten packen und anhängen
; - Photoscrap-Dateigröße korrigieren
;
:CREATE_PSCRAP  php                      ;Interrupt sperren.
                sei
;
                jsr    scrapCreateFile    ;Photoscrap-Datei erstellen.
                txa     ;Diskettenfehler?
                bne    :1                 ; => Ja, Abbruch...
;
                jsr    scrapDefSData      ;Startadr. Grafikdaten berechnen.
                jsr    scrapWrFile        ;Grafikdaten packen.
;
                LoadW  a0,COLOR_MATRIX   ;Startadresse der Farbdaten ab
                jsr    scrapDefCData      ;COLOR_MATRIX berechnen.
;
                jsr    scrapWrFile        ;Farbdaten packen.
                jsr    scrapClsFile       ;Letzten Block Photoscrap speicher.
;
:1              plp                      ;Interrupt-Status zurücksetzen.
                rts
```

```

;
; Zeiger auf Grafikdaten berechnen.
;
:scrapDefSData lda    scrapYPos          ;Grafilzeile 0-24 berechnen.
               lsr
               lsr
               lsr

               asl                    ;Anfangsadresse Grafikdaten im
               tax                    ;
               lda    dataStartGfx +0,x ;Speicher ab SCREEN_BASE für die
               clc                    ;aktuelle Grafikzeile berechnen.
               adc    #< SCREEN_BASE
               sta    a0L
               lda    dataStartGfx +1,x
               adc    #> SCREEN_BASE
               sta    a0H

               lda    scrapXPos        ;x-Koordinate in Cards nach
               asl                    ;Pixel umwandeln und zur Adresse
               asl                    ;der Grafikdaten addieren.
               asl

               php                    ;Überlauf im Carry-Flag speichern.
               clc
               adc    a0L
               sta    a0L
               plp                    ;"Add with Carry": Überlauf über

               lda    #$00            ;das Carry-Flag berücksichtigen.
               adc    a0H
               sta    a0H

               lda    scrapWidth      ;Anzahl Datenbyte in Grafikzeile:
               asl                    ;8 Byte je Card x Anzahl Cards
               asl
               asl
               sta    a4L

               ldx    #$00
               bcc    :1              ;Überlauf?
               inx                    ; => Ja, Highbyte anpassen.
               stx    a4H              ;Nur Werte von $0000-$013f möglich.

               lda    #< 40*8          ;Offset bis zum Beginn der
               sta    a5L              ;der nächsten Zeile festlegen.
               lda    #> 40*8
               sta    a5H

               lda    #8                ;Anzahl Daten innerhalb Zeile.
               sta    a3L

               rts

```

```

;
; Offset-Tabelle für Grafikdaten.
;
:dataStartGfx w 40 *8 *0 , 40 *8 *1 , 40 *8 *2 , 40 *8 *3
              w 40 *8 *4 , 40 *8 *5 , 40 *8 *6 , 40 *8 *7
              w 40 *8 *8 , 40 *8 *9 , 40 *8 *10 , 40 *8 *11
              w 40 *8 *12 , 40 *8 *13 , 40 *8 *14 , 40 *8 *15
              w 40 *8 *16 , 40 *8 *17 , 40 *8 *18 , 40 *8 *19
              w 40 *8 *20 , 40 *8 *21 , 40 *8 *22 , 40 *8 *23
              w 40 *8 *24

;
; Zeiger auf Farbdaten berechnen.
;
:scrapDefCData lda      scrapYPos          ;Grafilzeile 0-24 berechnen.
              lsr
              lsr
              lsr

              asl                      ;Anfangsadresse Farbdaten im
              tax                      ;Speicher ab COLOR_MATRIX für die
                                      ;aktuelle Farbzeile berechnen.

              lda      dataStartCol +0,x
              clc
              adc      #< COLOR_MATRIX
              sta      a0L
              lda      dataStartCol +1,x
              adc      #> COLOR_MATRIX
              sta      a0H

              lda      a0L              ;x-Koordinate zur Anfrangsadresse
                                      ;der Farbdaten addieren.
              clc
              adc      scrapXPos
              sta      a0L
              bcc      :1
              inc      a0H

::1          lda      scrapWidth        ;Anzahl Datenbyte in Farbzeile:
              sta      a4L              ;max. 40 Cards möglich.
              lda      #$00
              sta      a4H

              lda      #< 40            ;Offset bis zum Beginn der
              sta      a5L              ;der nächsten Zeile festlegen.
              lda      #> 40
              sta      a5H

              lda      #1                ;Anzahl Daten innerhalb Zeile.
              sta      a3L

              rts

```



```

;
; Offset-Tabelle für Farbdaten.
;
:dataStartCol w 40 *0 , 40 *1 , 40 *2 , 40 *3
              w 40 *4 , 40 *5 , 40 *6 , 40 *7
              w 40 *8 , 40 *9 , 40 *10 , 40 *11
              w 40 *12 , 40 *13 , 40 *14 , 40 *15
              w 40 *16 , 40 *17 , 40 *18 , 40 *19
              w 40 *20 , 40 *21 , 40 *22 , 40 *23
              w 40 *24

;
; Leere Photoscrap-Datei erzeugen.
;
; Dabei wird ein vorhandenes Photoscrap
; gelöscht und eine neue Datei mit den
; drei Headerbyte für die Größe des
; Photoscrap gespeichert.
;
:scrapCreateFile
    jsr      :delete          ;Vorhandene Datei löschen.

    lda      scrapWidth      ;Größe des Photoscrap in die
    sta      pScrapHdr +0    ;Headerbyte übernehmen.
    lda      scrapHeight
    sta      pScrapHdr +1
    lda      #$00            ;Höhe max. 200 Pixel, das
    sta      pScrapHdr +2    ;Highbyte ist daher immer NULL.

    LoadW   r9 ,HdrPS_Dok   ;Zeiger auf Infoblock.
    LoadB   r10L,$00        ;Zeiger auf Anfang Verzeichnis.
    jsr      SaveFile        ;Leeres Photoscrap speichern.
    txa
    bne      :delete        ;Diskettenfehler?
                                ; => Ja, Abbruch...

    LoadW   r6,photoScrapName
    jsr      FindFile        ;Dateieintrag Photoscrap suchen.
    txa
    bne      :delete        ;Diskettenfehler?
                                ; => Ja, Abbruch...

    lda      r1L             ;Adresse des Verzeichnisblock
    sta      dirEntryBlk +0  ;zwischenspeichern.
    lda      r1H
    sta      dirEntryBlk +1

    lda      r5L             ;Zeiger auf Eintrag innerhalb
    sta      dirEntryAdr +0  ;des Verzeichnisblock speichern.
    lda      r5H
    sta      dirEntryAdr +1

    ldx      #1              ;Suche für nächsten freien
    stx      nextFreeBlk +0  ;Block initialisieren.
    dex
    stx      nextFreeBlk +1

```

```

        lda    dirEntryBuf +1    ;Zeiger auf Track/Sektor des ersten
        ldx    dirEntryBuf +2    ;Block im Photoscrap einlesen.

        sta    curBlock +0       ;Adresse des aktuellen Datenblock
        stx    curBlock +1       ;zwischenspeichern.

        ldy    #5 -1             ;Zeiger auf das letzte Byte im
        sty    curByte           ;aktuellen Datenblock definieren.

        sta    r1L               ;Zeiger auf Track/Sektor des
        stx    r1H               ;ersten Datenblock im Photoscrap.

        jsr    GetBlockBuf        ;Ersten Block Photoscrap einlesen.
        txa                     ;Diskettenfehler?
        beq    :done             ; => Nein, Ende...

::delete    LoadW    r0,photoScrapName
        jsr    DeleteFile        ;Vorhandenes Photoscrap löschen.
        ;                      ;Diskettenfehler?
        ;                      ;
        beq    :done             ; => Nein, Ende...

::done      rts

;
;  Daten in Photoscrap-Datei speichern.
;
;  Dabei werden die Grafik-/Farbdaten
;  zeilenweise in den Zwischenspeicher
;  übertragen, gepackt und dann an die
;  Photoscrap-Datei angehängt.
;
:scrapWrFile    lda    scrapHeight    ;Anzahl Zeilen in Cards
                lsr                    ;berechnen.
                lsr
                lsr

;
;  Test auf Anzahl Zeilen=0 kann
;  entfallen, da Photoscrap mindestens
;  1 Card hoch ist.
;
::loop        pha                    ;Zähler auf Stack speichern.

                jsr    scrapCopyData    ;Daten einlesen.

                jsr    scrapPackData    ;Daten packen.

                jsr    scrapUpdFile     ;Gepackte Daten speichern.

                pla                    ;Zeilen-Zähler wieder einlesen.

                cpx    #$00            ;Diskettenfehler?
                bne    :err            ; => Ja, Abbruch...

```

```

                sec                ;Zeilen-Zähler korrigieren.
                sbc    #1          ;Alle Zeilen gespeichert?
                bne     :loop      ; => Nein, weiter...

::err          rts

;
;  Daten an Photoscrap anhängen.
;
;  Ablauf:
;  - Test auf "Daten vorhanden"
;  - Ist Datenblock voll?
;    - Ja, Datenblock speichern und
;      neuen Datenblock anlegen.
;    - Nein, Byte in Datenblock
;      übernehmen.
;  - Weiter bis alle Byte gespeichert.
;
:scrapUpdFile  lda    #< dataPacked ;Zeiger auf gepackte Daten.
                sta    r0L
                lda    #> dataPacked
                sta    r0H

::next         lda    r2L           ;Sind noch Daten vorhanden?
                ora    r2H
                bne    :1           ; => Ja, weiter...

                ldx    #$00         ;Ende, kein Fehler.

::err          rts

::1            ldx    curByte       ;Zeiger auf letztes Byte einlesen.
                inx                ;Ist aktueller Datenblock voll?
                bne    :2           ; => Nein, weiter...

                jsr    GetNxFreeBlk ;Freien Block suchen.
                txa                ;Diskettenfehler?
                bne    :err         ; => Ja, Abbruch...

                ldx    #2           ;Zeiger auf nächste Byte-Position.

                inc    dirEntryBuf +28 ;Anzahl Blocks für Photoscrap
                bne    :2           ;korrigieren.
                inc    dirEntryBuf +29

::2            stx    curByte       ;Zeiger auf Datenbyte speichern.

                ldy    #$00         ;Wert aus Zwischenspeicher
                lda    (r0L),y      ;einlesen und in Datenblock
                sta    diskBlkBuf,x ;übernehmen.

                lda    r2L           ;Anzahl Datenbytes -1.
                bne    :3
                dec    r2H

::3            dec    r2L

```

```

        inc    r0L                ;Zeiger auf Zwischenspeicher
        bne    :4                ;korrigieren.
        inc    r0H
::4      jmp    :next            ;Weiter mit nächstem Byte.

;
; Photoscrap-Datei schließen.
;
; Dabei wird der letzte Datenblock der
; noch im Speicher ist auf Diskette
; gespeichert und die Blockanzahl für
; das Photoscrap korrigiert.
;
;
:scrapClsFile  lda    #$00        ;Linkbyte im aktuellen Datenblock
               sta    diskBlkBuf +0 ;auf $00=Dateiende setzen.
               lda    curByte     ;Zeiger auf das letzte Byte im
               sta    diskBlkBuf +1 ;aktuellen Datenblock übernehmen.

               lda    curBlock +0 ;Adresse des aktuellen
               sta    r1L          ;Datenblock einlesen.
               lda    curBlock +1
               sta    r1H

               jsr    PutBlockBuf  ;Aktuellen Block speichern.
               txa                ;Diskettenfehler?
               bne    :err         ; => Ja, Abbruch...

               lda    dirEntryBlk +0 ;Track/Sektor für Verzeichnisblock
               sta    r1L          ;des Photoscrap einlesen.
               lda    dirEntryBlk +1
               sta    r1H

               jsr    GetBlock     ;Verzeichnisblock einlesen.
               txa                ;Diskettenfehler?
               bne    :err         ; => Ja, Abbruch...

               ldy    dirEntryAdr  ;Anzahl Blocks für Photoscrap
               lda    dirEntryBuf +28 ;im Verzeichniseintrag anpassen.
               sta    diskBlkBuf +28,y
               lda    dirEntryBuf +29
               sta    diskBlkBuf +29,y

               jsr    PutBlock     ;Verzeichnisblock speichern.
               txa                ;Diskettenfehler?
               bne    :err         ; => Ja, Abbruch...

;
; SetNextFree reserviert den nächsten
; Block nur in der BAM im Speicher.
; Zum Schluss die BAM speichern!
;
;
               jsr    PutDirHead  ;BAM aktualisieren.
               txa                ;Diskettenfehler?
               bne    :err         ; => Ja, Abbruch...
::err      rts

```

```

;
; Sektor in diskBlkBuf schreiben
;
;
:PutBlockBuf   lda    #< diskBlkBuf      ;Zeiger auf Zwischenspeicher für
               sta    r4L                ;den aktuellen Datenblock setzen.
               lda    #> diskBlkBuf
               sta    r4H

               jmp     PutBlock          ;Aktuellen Block speichern.

;
; Sektor nach diskBlkBuf einlesen
;
;
:GetBlockBuf   lda    #< diskBlkBuf      ;Zeiger auf Zwischenspeicher für
               sta    r4L                ;den aktuellen Datenblock setzen.
               lda    #> diskBlkBuf
               sta    r4H

               jmp     GetBlock          ;Aktuellen Block speichern.

;
; Nächsten freien Block suchen
;
;
:GetNxFreeBlk  lda    nextFreeBlk +0
               sta    r3L
               lda    nextFreeBlk +1
               sta    r3H
               jsr     SetNextFree       ;Neuen freien Datenblock suchen.
               txa                      ;Diskettenfehler?
               bne     :err              ; => Ja, Abbruch...

               lda    curBlock +0        ;Adresse des aktuellen
               sta    r1L                ;Datenblock einlesen.
               lda    curBlock +1
               sta    r1H

               lda    r3L                ;Adresse des freien Datenblock.
               ldx    r3H
               sta    nextFreeBlk +0    ;Adresse Track/Sektor als neue
               stx    nextFreeBlk +1    ;Startwert für Sektorsuche setzen.

               sta    curBlock +0        ;Neuer Datenblock als "Aktuell"
               stx    curBlock +1        ;setzen.

               sta    diskBlkBuf +0     ;Adresse des neuen Datenblock als
               stx    diskBlkBuf +1     ;Linkbytes setzen.

               jsr     PutBlockBuf       ;Aktuellen Block speichern.
               txa                      ;Diskettenfehler?
               bne     :err              ; => Ja, Abbruch...

::err          rts

```

```

;
; Photoscrap-Datei erstellen.
;
;
; Dazu wird ein 40Z-Bildausschnitt mit
; Grafik+Farbe in eine PhotoScrap-Datei
; gespeichert.
;
; Übergabe:
:scrapXPos      b $01 ;x-Position in Cards.
:scrapYPos      b $08 ;y-Position, nur ganze 8er-Blöcke!
:scrapWidth     b $03 ;Breite: In Cards!
:scrapHeight    b $18 ;Höhe : Nur ganze 8er-Blöcke!
;
; Interne Variablen:
:dirEntryBlk    b $00,$00
:dirEntryAdr    w $0000
:curBlock       b $00,$00
:curByte        b $00
:nextFreeBlk    b $00,$00
;
; Benötigter Datenspeicher:
; :dataUnpacked = max. 40x8 Byte ungepackte Daten.
; :dataPacked   = max. 40x8 Byte + ca.30 Kompressionsbyte, wenn alle
;               Datenbyte verschieden sind und ein packen unmöglich ist.
;
;
; PhotoScrap erstellen.
;
; Hauptroutine:
; - Leere Photoscrap-Datei erstellen
; - Grafikdaten packen und anhängen
; - Farbdaten packen und anhängen
; - Photoscrap-Dateigröße korrigieren
;
:CREATE_PSCRAP php                      ;Interrupt sperren.
                sei
;
;               jsr    scrapCreateFile  ;Photoscrap-Datei erstellen.
;               txa    ;Diskettenfehler?
;               bne    :1                ; => Ja, Abbruch...
;
;               jsr    scrapDefScrData  ;Startadr. Grafikdaten berechnen.
;
;               jsr    scrapWriteFile   ;Grafikdaten packen.
;
;               LoadW a0,COLOR_MATRIX ;Startadresse der Farbdaten ab
;               jsr    scrapDefColData  ;COLOR_MATRIX berechnen.
;
;               jsr    scrapWriteFile   ;Farbdaten packen.
;
;               jsr    scrapCloseFile   ;Letzten Block Photoscrap speicher.
;
::1            plp                      ;Interrupt-Status zurücksetzen.
                rts

```

```

;
; Zeiger auf Grafikdaten berechnen.
;
:scrapDefScrData

        lda    scrapYPos          ;Grafilzeile 0-24 berechnen.
        lsr
        lsr
        lsr

        asl
        tax                      ;Anfangsadresse Grafikdaten im

        lda    dataStartGfx +0,x ;Speicher ab SCREEN_BASE für die
        clc                                ;aktuelle Grafikzeile berechnen.
        adc    #< SCREEN_BASE
        sta    a0L
        lda    dataStartGfx +1,x
        adc    #> SCREEN_BASE
        sta    a0H

        lda    scrapXPos          ;x-Koordinate in Cards nach
        asl                                ;Pixel umwandeln und zur Adresse
        asl                                ;der Grafikdaten addieren.
        asl
        php                                ;Überlauf im Carry-Flag speichern.
        clc
        adc    a0L
        sta    a0L
        plp                                ;"Add with Carry": Überlauf über
        lda    #$00                      ;das Carry-Flag berücksichtigen.
        adc    a0H
        sta    a0H

        lda    scrapWidth          ;Anzahl Datenbyte in Grafikzeile:
        asl                                ;8 Byte je Card x Anzahl Cards
        asl
        asl
        sta    a4L

        ldx    #$00
        bcc    :1                    ;Überlauf?
        inx                                ; => Ja, Highbyte anpassen.
        stx    a4H                    ;Nur Werte von $0000-$013f möglich.

        lda    #< 40*8                ;Offset bis zum Beginn der
        sta    a5L                    ;der nächsten Zeile festlegen.
        lda    #> 40*8
        sta    a5H

        lda    #8                    ;Anzahl Daten innerhalb Zeile.
        sta    a3L

        rts

```

```

;
; Offset-Tabelle für Grafikdaten.
;
:dataStartGfx w 40 *8 *0 , 40 *8 *1 , 40 *8 *2 , 40 *8 *3
              w 40 *8 *4 , 40 *8 *5 , 40 *8 *6 , 40 *8 *7
              w 40 *8 *8 , 40 *8 *9 , 40 *8 *10 , 40 *8 *11
              w 40 *8 *12 , 40 *8 *13 , 40 *8 *14 , 40 *8 *15
              w 40 *8 *16 , 40 *8 *17 , 40 *8 *18 , 40 *8 *19
              w 40 *8 *20 , 40 *8 *21 , 40 *8 *22 , 40 *8 *23
              w 40 *8 *24

;
; Zeiger auf Farbdaten berechnen.
;
:scrapDefColData
                lda    scrapYPos          ;Grafilzeile 0-24 berechnen.
                lsr
                lsr
                lsr

                asl
                tax                        ;Anfangsadresse Farbdaten im

                lda    dataStartCol +0,x ;Speicher ab COLOR_MATRIX für die
                clc                                ;aktuelle Farbzeile berechnen.
                adc    #< COLOR_MATRIX
                sta    a0L
                lda    dataStartCol +1,x
                adc    #> COLOR_MATRIX
                sta    a0H

                lda    a0L                    ;x-Koordinate zur Anfrangsadresse
                clc                                ;der Farbdaten addieren.
                adc    scrapXPos
                sta    a0L
                bcc    :1
                inc    a0H

::1            lda    scrapWidth            ;Anzahl Datenbyte in Farbzeile:
                sta    a4L                    ;max. 40 Cards möglich.
                lda    #$00
                sta    a4H

                lda    #< 40                ;Offset bis zum Beginn der
                sta    a5L                    ;der nächsten Zeile festlegen.
                lda    #> 40
                sta    a5H

                lda    #1                    ;Anzahl Daten innerhalb Zeile.
                sta    a3L

                rts

```



```

;
; Offset-Tabelle für Farbdaten.
;
:dataStartCol w 40 *0 , 40 *1 , 40 *2 , 40 *3
              w 40 *4 , 40 *5 , 40 *6 , 40 *7
              w 40 *8 , 40 *9 , 40 *10 , 40 *11
              w 40 *12 , 40 *13 , 40 *14 , 40 *15
              w 40 *16 , 40 *17 , 40 *18 , 40 *19
              w 40 *20 , 40 *21 , 40 *22 , 40 *23
              w 40 *24

;
; Leere Photoscrap-Datei erzeugen.
;
; Dabei wird ein vorhandenes Photoscrap
; gelöscht und eine neue Datei mit den
; drei Headerbyte für die Größe des
; Photoscrap gespeichert.
;
:scrapCreateFile

        jsr      :delete          ;Vorhandene Datei löschen.

        lda      scrapWidth       ;Größe des Photoscrap in die
        sta      pScrapHdr +0     ;Headerbyte übernehmen.
        lda      scrapHeight
        sta      pScrapHdr +1
        lda      #$00             ;Höhe max. 200 Pixel, das
        sta      pScrapHdr +2     ;Highbyte ist daher immer NULL.

        LoadW   r9 ,HdrPS_Dok    ;Zeiger auf Infoblock.
        LoadB   r10L,$00         ;Zeiger auf Anfang Verzeichnis.
        jsr      SaveFile        ;Leeres Photoscrap speichern.
        txa      ;Diskettenfehler?
        bne      :delete         ; => Ja, Abbruch...

        LoadW   r6,photoScrapName
        jsr      FindFile        ;Dateieintrag Photoscrap suchen.
        txa      ;Diskettenfehler?
        bne      :delete         ; => Ja, Abbruch...

        lda      r1L              ;Adresse des Verzeichnisblock
        sta      dirEntryBlk +0  ;zwischenspeichern.
        lda      r1H
        sta      dirEntryBlk +1

        lda      r5L              ;Zeiger auf Eintrag innerhalb
        sta      dirEntryAdr +0  ;des Verzeichnisblock speichern.
        lda      r5H
        sta      dirEntryAdr +1

        ldx      #1               ;Suche für nächsten freien
        stx      nextFreeBlk +0  ;Block initialisieren.
        dex
        stx      nextFreeBlk +1

```

```

        lda    dirEntryBuf +1    ;Zeiger auf Track/Sektor des ersten
        ldx    dirEntryBuf +2    ;Block im Photoscrap einlesen.

        sta    curBlock +0       ;Adresse des aktuellen Datenblock
        stx    curBlock +1       ;zwischenspeichern.

        ldy    #5 -1             ;Zeiger auf das letzte Byte im
        sty    curByte           ;aktuellen Datenblock definieren.

        sta    r1L               ;Zeiger auf Track/Sektor des
        stx    r1H               ;ersten Datenblock im Photoscrap.

        jsr    GetBlockBuf       ;Ersten Block Photoscrap einlesen.
        txa                     ;Diskettenfehler?
        beq    :done             ; => Nein, Ende...

::delete    LoadW    r0,photoScrapName
        jsr    DeleteFile        ;Vorhandenes Photoscrap löschen.
        ;                      ;Diskettenfehler?
        ;                      ;
        beq    :done             ; => Nein, Ende...

::done      rts

;
; Daten in Photoscrap-Datei speichern.
;
; Dabei werden die Grafik-/Farbdaten
; zeilenweise in den Zwischenspeicher
; übertragen, gepackt und dann an die
; Photoscrap-Datei angehängt.
;
:scrapWriteFile

        lda    scrapHeight       ;Anzahl Zeilen in Cards
        lsr                     ;berechnen.
        lsr
        lsr

;
; Test auf Anzahl Zeilen=0 kann
; entfallen, da Photoscrap mindestens
; 1 Card hoch ist.
;
::loop      pha                     ;Zähler auf Stack speichern.

        jsr    scrapCopyData      ;Daten einlesen.

        jsr    scrapPackData      ;Daten packen.

        jsr    scrapUpdateFile    ;Gepackte Daten speichern.

        pla                     ;Zeilen-Zähler wieder einlesen.

        cpx    #$00              ;Diskettenfehler?
        bne    :err              ; => Ja, Abbruch...

```

```

                sec                ;Zeilen-Zähler korrigieren.
                sbc    #1          ;Alle Zeilen gespeichert?
                bne     :loop      ; => Nein, weiter...

::err           rts

;
; Daten an Photoscrap anhängen.
;
; Ablauf:
; - Test auf "Daten vorhanden"
; - Ist Datenblock voll?
;   - Ja, Datenblock speichern und
;     neuen Datenblock anlegen.
;   - Nein, Byte in Datenblock
;     übernehmen.
; - Weiter bis alle Byte gespeichert.
;
:scrapUpdateFile

                lda    #< dataPacked ;Zeiger auf gepackte Daten.
                sta    r0L
                lda    #> dataPacked
                sta    r0H

::next         lda    r2L            ;Sind noch Daten vorhanden?
                ora    r2H
                bne    :1            ; => Ja, weiter...

                ldx    #$00          ;Ende, kein Fehler.
::err          rts

::1            ldx    curByte        ;Zeiger auf letztes Byte einlesen.
                inx
                bne    :2            ;Ist aktueller Datenblock voll?
                ; => Nein, weiter...

                jsr    GetNxFreeBlk  ;Freien Block suchen.
                txa
                bne    :err          ;Diskettenfehler?
                ; => Ja, Abbruch...

                ldx    #2            ;Zeiger auf nächste Byte-Position.

                inc    dirEntryBuf +28 ;Anzahl Blocks für Photoscrap
                bne    :2            ;korrigieren.
                inc    dirEntryBuf +29

::2            stx    curByte        ;Zeiger auf Datenbyte speichern.

                ldy    #$00          ;Wert aus Zwischenspeicher
                lda    (r0L),y       ;einlesen und in Datenblock
                sta    diskBlkBuf,x  ;übernehmen.

                lda    r2L            ;Anzahl Datenbytes -1.
                bne    :3
                dec    r2H
                dec    r2L
::3

```

```

        inc    r0L            ;Zeiger auf Zwischenspeicher
        bne    :4            ;korrigieren.
        inc    r0H

:4      jmp     :next        ;Weiter mit nächstem Byte.

;
; Photoscrap-Datei schließen.
;
; Dabei wird der letzte Datenblock der
; noch im Speicher ist auf Diskette
; gespeichert und die Blockanzahl für
; das Photoscrap korrigiert.
;
;scrapCloseFile

        lda    #$00          ;Linkbyte im aktuellen Datenblock
        sta    diskBlkBuf +0 ;auf $00=Dateiende setzen.
        lda    curByte       ;Zeiger auf das letzte Byte im
        sta    diskBlkBuf +1 ;aktuellen Datenblock übernehmen.

        lda    curBlock +0   ;Adresse des aktuellen
        sta    r1L           ;Datenblock einlesen.
        lda    curBlock +1
        sta    r1H

        jsr    PutBlockBuf   ;Aktuellen Block speichern.
        txa                 ;Diskettenfehler?
        bne    :err          ; => Ja, Abbruch...

        lda    dirEntryBlk +0 ;Track/Sektor für Verzeichnisblock
        sta    r1L           ;des Photoscrap einlesen.
        lda    dirEntryBlk +1
        sta    r1H

        jsr    GetBlock      ;Verzeichnisblock einlesen.
        txa                 ;Diskettenfehler?
        bne    :err          ; => Ja, Abbruch...

        ldy    dirEntryAdr    ;Anzahl Blocks für Photoscrap
        lda    dirEntryBuf +28 ;im Verzeichniseintrag anpassen.
        sta    diskBlkBuf +28,y
        lda    dirEntryBuf +29
        sta    diskBlkBuf +29,y

        jsr    PutBlock      ;Verzeichnisblock speichern.
        txa                 ;Diskettenfehler?
        bne    :err          ; => Ja, Abbruch...

;
; SetNextFree reserviert den nächsten
; Block nur in der BAM im Speicher.
; Zum Schluss die BAM speichern!
;

```

```

        jsr    PutDirHead      ;BAM aktualisieren.
;      txa      ;Diskettenfehler?
;      bne     :err          ; => Ja, Abbruch...

::err      rts

;
; Sektor in diskBlkBuf schreiben
;
;
PutBlockBuf  lda    #< diskBlkBuf      ;Zeiger auf Zwischenspeicher für
        sta     r4L                  ;den aktuellen Datenblock setzen.
        lda     #> diskBlkBuf
        sta     r4H

        jmp     PutBlock              ;Aktuellen Block speichern.

;
; Sektor nach diskBlkBuf einlesen
;
;
GetBlockBuf  lda    #< diskBlkBuf      ;Zeiger auf Zwischenspeicher für
        sta     r4L                  ;den aktuellen Datenblock setzen.
        lda     #> diskBlkBuf
        sta     r4H

        jmp     GetBlock              ;Aktuellen Block speichern.

;
; Nächsten freien Block suchen
;
;
GetNxFreeBlk  lda    nextFreeBlk +0
        sta     r3L
        lda     nextFreeBlk +1
        sta     r3H

        jsr     SetNextFree          ;Neuen freien Datenblock suchen.
        txa      ;Diskettenfehler?
        bne     :err                ; => Ja, Abbruch...

        lda     curBlock +0          ;Adresse des aktuellen
        sta     r1L                  ;Datenblock einlesen.
        lda     curBlock +1
        sta     r1H

        lda     r3L                  ;Adresse des freien Datenblock.
        ldx     r3H
        sta     nextFreeBlk +0      ;Adresse Track/Sektor als neue
        stx     nextFreeBlk +1      ;Startwert für Sektorsuche setzen.

        sta     curBlock +0          ;Neuer Datenblock als "Aktuell"
        stx     curBlock +1          ;setzen.

        sta     diskBlkBuf +0        ;Adresse des neuen Datenblock als
        stx     diskBlkBuf +1        ;Linkbytes setzen.

```

```

;          jsr    PutBlockBuf    ;Aktuellen Block speichern.
;          txa                     ;Diskettenfehler?
;          bne     :err          ; => Ja, Abbruch...

::err      rts

;
; Photoscrap-Daten kopieren.
;
; Die Daten werden aus dem Bildschirm-
; oder Farbspeicher zuerst ungepackt in
; den Zwischenspeicher kopiert.
; Dabei werden Grafikdaten in ganzen
; Pixelzeilen kopiert. Das ganze wird
; für 8 Pixelzeilen wiederholt.
; Bei Farbdaten wird nur eine Zeile
; in den Zwischenspeicher kopiert.
;
; Übergabe:
; a0 = Zeiger auf ungepackte Daten.
; a4 = Anzahl zu packender Daten.
; a5 = Offset zur nächsten Zeile.
; a3L = Anzahl Pixelzeilen.
;
:scrapCopyData lda    a0L          ;Zeiger auf ungepackte Daten.
               ldx    a0H
               sta    r0L
               stx    r0H

               lda    #< dataUnpacked ;Zeiger auf Zwischenspeicher.
               ldx    #> dataUnpacked
               sta    r1L
               stx    r1H

               lda    #0            ;Zähler für Pixelzeilen
               sta    a3H            ;initialisieren.

::1          ldx    scrapWidth      ;Anzahl Cards in Zeile einlesen.

               lda    r0H            ;Zeiger auf Anfang der aktuellen
               pha                     ;Zeile zwischenspeichern.
               lda    r0L
               pha

::2          ldy    #0            ;Grafik- oder Farbbyte kopieren.
               lda    (r0L),y
               sta    (r1L),y

               lda    r0L            ;Zeiger auf nächstes Byte in
               clc                     ;aktueller Zeile berechnen.
               adc    a3L
               sta    r0L
               bcc    :3
               inc    r0H

```

```

::3      inc    r1L          ;Zeiger auf nächstes Byte im
      bne     :4            ;Zwischenspeicher setzen.
      inc     r1H

::4      dex          ;Alle Cards bearbeitet?
      bne     :2            ; => Nein, weiter...

      pla          ;Startadresse der nächsten
      clc          ;Pixelzeile berechnen.
      adc     #< 1
      sta     r0L
      pla
      adc     #> 1
      sta     r0H

      inc     a3H          ;Zeilenzähler +1.

      lda     a3H          ;Wurde alle Pixelzeilen bzw. die
      cmp     a3L          ;komplette Farbzeile kopiert?
      bne     :1            ; => Nein, weiter...

      Addw    a5,a0        ;Zeiger auf nächste Zeile.

      rts

;
; Daten packen.
;
; dataUnpacked = Ungepackte Daten.
; dataPacked = Zwischenspeicher.
;
; Verwendete Register:
; a4 = Anzahl ungepackte Datenbyte.
; a6 = Anzahl der noch zu bearbeitenden Bytes.
; a9L = Anzahl identische Bytes.
; a9H = Anzahl ungepackter Bytes.
;
; Rückgabe:
; r2 = Anzahl gepackte Datenbyte.
;
:scrapPackData LoadW r0,dataUnpacked ;Zeiger auf ungepackte Daten.
      LoadW r1,dataPacked ;Zeiger auf Zwischenspeicher.

      Movew   a4,a6        ;Anzahl Bytes in Zeile.

      lda     #$00         ;Anzahl identische Byte
      sta     a9L          ;zurücksetzen.

;*** Bytes aus Zwischenspeicher einlesen, packen und in Speicher für
; GeoPaint-Datensatz kopieren.
::next    jsr     scrapEqualBytes ;Nach gleichen Bytes suchen.

      ldy     #< scrapPackNone ;Vorgabe:
      ldx     #> scrapPackNone ;Daten nicht packen.

```

```

        lda    a9L                ;Anzahl zu packender Bytes.
        cmp    #$04              ;Mehr als vier Bytes?
        bcc    :1                ; => Ja, Daten nicht packen.

        ldy    #< scrapPackBytes ;$8x=Einzelbyte (max. 127x) packen.
        ldx    #> scrapPackBytes

:::1    tya
        jsr    CallRoutine        ;Daten packen/nicht packen.

        lda    a6L
        ora    a6H                ;Alle Bytes gepackt?
        bne    :next              ; => Nein, weiter...

        lda    r1L                ;Anzahl der gepackten Datenbyte
        sec                                ;berechnen.
        sbc    #< dataPacked
        sta    r2L
        lda    r1H
        sbc    #> dataPacked
        sta    r2H
        rts

;
; Identische Datenbyte suchen.
;
; Sucht in den ungepackten Datenbyte
; mehrere gleiche, aufeinanderfolgende
; Einzelbyte.
;
;:scrapEqualBytes

        lda    a9L                ;Sind noch gleiche Einzelbytes
        bne    :exit              ;im Speicher? Nein, Daten noch
                                ;nicht komplett gepackt, nächste
                                ;Einzelbytes packen.

        ldy    #$00              ;Zeiger auf aktuelles Byte.
        lda    (r0L),y            ;Aktuelles Byte einlesen.
        iny                                ;Zeiger auf nächstes Byte.
:::loop    cmp    (r0L),y          ;Byte identisch mit aktuellem Byte?
        bne    :done              ; => Nein, weiter...
        iny                                ;Zähler für gleiche Byte erhöhen.
        cpy    #$7f              ;Max. 127 gleiche Bytes erreicht?
        bcc    :loop              ; => Nein, weiter...

:::done    lda    a6H                ;Anzahl gleiche Bytes mit Anzahl
        bne    :1                ;der noch zu packenden Bytes
        cpy    a6L                ;vergleichen.
        bcc    :1
        beq    :1
        ldy    a6L                ;Anzahl Bytes auf Restbytes setzen.

:::1    tya
        sta    a9L                ;Anzahl gleicher Einzelbytes
:::exit    rts                    ;zwischenspeichern.

```



```

;
; Gleiche Einzelbytes packen.
;
; Übergabe:
; a9L = Anzahl gleiche Datenbyte.
; r0 = Zeiger auf ungepackte Daten.
; r1 = Zeiger auf gepackte Daten.
;
:scrapPackBytes

        lda    a9L                ;Anzahl Einzelbytes einlesen.
        ldy    #$00                ;Zeiger auf Zwischenpeicher.
        sta    (r1L),y            ;Kompressionsbyte $01-$7f setzen.

        lda    (r0L),y            ;Zu packendes Datenbyte einlesen.
        iny
        sta    (r1L),y            ;Zeiger auf nächstes Byte setzen.
        ;Byte in Zwischenpeicher kopieren.

        lda    #2                  ;Zeiger für Zwischenpeicher auf
        clc                        ;nächstes Byte setzen.
        adc    r1L
        sta    r1L
        bcc    :1
        inc    r1H

::1      lda    a9L                ;Zeiger auf Datenbyte um Anzahl
        clc                        ;der Einzelbytes erhöhen.
        adc    r0L
        sta    r0L
        bcc    :2
        inc    r0H

::2      lda    a6L                ;Anzahl noch zu packender Bytes
        sec                        ;korrigieren.
        sbc    a9L                ; => In :a9L steht die Anzahl der
        sta    a6L                ; gepackten Einzelbytes.
        bcs    :3
        dec    a6H

::3      lda    #$00                ;Zähler Anzahl Einzelbyte löschen.
        sta    a9L

        rts

;
; Daten ungepackt übernehmen.
;
; Übergabe:
; r0 = Zeiger auf ungepackte Daten.
; r1 = Zeiger auf gepackte Daten.
;
:scrapPackNone jsr    scrapCountBytes ;Ungepackte Bytes zählen.

```

```

        lda    a9H                ;Anzahl ungepackter Bytes.
        cmp    #($dc-$81)         ;Mehr als 90 Byte?
        bcc    :1                 ; => Nein, weiter...
        lda    #($dc-$81)         ;Max. 90 ungepackte Byte möglich.
        sta    a9H
:::1    ora    #%10000000         ;Packmodus "Ungepackt" setzen.
        ldz    #$00               ;Zeiger auf Zwischenspeicher.
        stz    (r1L),y           ;Kompressionsbyte speichern.

        inc    r1L
        bne    :2
        inc    r1H

:::2    ldz    a9H                ;Anzahl ungepackter Bytes in
        dey    ;Zwischenspeicher kopieren.
        jsr    scrapCopyYRegByt

        lda    a9H                ;Zeiger für Zwischenspeicher auf
        clc                     ;nächstes Byte setzen.
        adc    r1L
        stz    r1L
        bcc    :3
        inc    r1H

:::3    lda    a9H                ;Zeiger auf Datenbyte um Anzahl
        clc                     ;ungepackter Bytes erhöhen.
        adc    r0L
        stz    r0L
        bcc    :4
        inc    r0H

:::4    lda    a6L                ;Anzahl noch zu packender Bytes
        sec                     ;korrigieren.
        sbc    a9H                ; => In :a9H steht die Anzahl der
        sta    a6L                ;   ungepackten Einzelbytes.
        bcs    :5
        dec    a6H

:::5    rts

;
; Datenbytes kopieren.
;
; Übergabe:
; yReg = Anzahl Bytes -1
;       max. 128 Byte!
;
; scrapCopyYRegByt

        lda    (r0L),y           ;Byte einlesen und in
        stz    (r1L),y           ;Zwischenspeicher kopieren.
        dey    ;Alle Bytes kopiert?
        bpl    scrapCopyYRegByt ; => Nein, weiter...
        rts

```

```

;
; Anzahl ungepackter Daten berechnen.
;
; Rückgabe:
; a9H = Anzahl Einzelbytes.
;
:scrapCountBytes

        lda    #$01                ;Max. Anzahl ungepackter Bytes auf
        sta    a9H                ;Startwert setzen.

        PushW  r0                  ;Zeiger auf Grafikdaten retten.
        PushW  a6                  ;Anzahl zu packender Bytes retten.

        jsr    scrapPosNxByte      ;Zeiger auf nächstes Byte setzen.

::loop   lda    a6L                ;Weitere Bytes in Grafikspeicher
        ora    a6H                ;zum packen vorhanden?
        beq    :exit              ; => Nein, Ende...

        lda    #$00                ;Einzelbyte-Flag löschen.
        sta    a9L

        jsr    scrapEqualBytes     ;Gleiche Einzelbytes suchen.
        cmp    #4                  ;Mehr als vier gleiche Bytes?
        bcs    :exit              ; => Ja, Einzelbytes packen.

        jsr    scrapPosNxByte      ;Zeiger auf nächstes Byte.

        inc    a9H                ;Anzahl ungepackter Datenbyte +1.

        lda    a9H
        cmp    #90 +1             ;Max. 90 Bytes gefunden?
        bcc    :loop              ; => Nein weiter...

        lda    #$00                ;Einzelbyte-Flag löschen.
        sta    a9L

::exit   PopW   a6                  ;Anzahl noch zu packender Bytes
        PopW   r0                  ;und Zeiger auf Grafikdaten wieder

        rts                       ;zurücksetzen.

;
; Zeiger auf nächstes Datenbyte.
;
; Übergabe:
; r0 = Zeiger Originaldaten.
; a6 = Zähler Restdaten.
;
:scrapPosNxByte

        inc    r0L                ;Zeiger auf nächstes Byte der
        bne    :51                ;Grafikdaten setzen.
        inc    r0H


```

```

::51      lda    a6L            ;Anzahl noch zu packender Bytes
          bne    :52            ;korrigieren.
          dec    a6H
::52      dec    a6L

          rts

;
; Name der Photoscrap-Datei.
;
:photoScrapName
          b "Photo Scrap",NULL

;
; Infoblock für Photoscrap-Datei.
;
:HdrPS_Dok    w photoScrapName
              b $03,$15
              j

:HdrPS_068    b $00!SEQ
:HdrPS_069    b SYSTEM
:HdrPS_070    b SEQUENTIAL
:HdrPS_071    w pScrapHdr            ;Zeiger auf Header-Bytes.
              w pScrapHdr +3        ;Nur 3 Byte speichern.
              w $0000
:HdrPS_077    b "Photo Scrap "      ;Klasse.
:HdrPS_089    b "V1.1"              ;Version.
:HdrPS_093    b $00,$00,$00,$00    ;Reserviert.
:HdrPS_160    e HdrPS_Dok +160 +1   ;Info.

;
; Headerbytes für Photoscrap:
; 1 Byte = Breite in Cards.
; 1 Word = Höhe in Pixel.
;
:pScrapHdr    b $00                ;Breite in Cards.
              w $0000              ;Höhe in Pixel.

```

Zum Schluss noch die Routinen zum schreiben eines GeoPaint-Dokumentes:

```

;
; Screenshot erstellen.
;
; Dazu wird ein 40Z-Bildschirm mit
; Grafik+Farbe in einer GeoPaint-Datei
; gespeichert.
;
; Benötigter Datenspeicher:
; :dataUnpacked = 1280 +8 +160 +1 Byte ungepackte Daten.
; :dataPacked   = 1280 +8 +160 +1 +48 Byte gepackte Daten.

```

```

;
; Im ungünstigsten Fall müssen alle
; Grafikdaten ungepackt gespeichert
; werden. In dem Fall werden zusätzlich
; ca.48Bytes (1448 Daten / max.31Bytes)
; für :dataPacked benötigt.
;
:CREATE_GIMAGE php                                ;Interrupt sperren.
sei

        jsr    paintCreateFile    ;GeoPaint-Datei erstellen.
        txa                                ;Diskettenfehler?
        bne     :err                ; => Ja, Abbruch...

        LoadW  r0,paintFileName
        jsr    OpenRecordFile      ;GeoPaint-Datei öffnen.
        txa                                ;Diskettenfehler?
        bne     :err                ; => Ja, Abbruch...

        jsr    paintWriteFile      ;ScreenShot erstellen.
        txa                                ;Diskettenfehler?
        bne     :err                ; => Ja, Abbruch...

        jsr    UpdateRecordFile    ;VLIR-Datei aktualisieren und
        jsr    CloseRecordFile     ;GeoPaint-Dokument schließen.
        txa                                ;Diskettenfehler?
        beq     :1                  ; => Nein, Ende...

;
; Disk-I/O-Fehler.
; Unvollständige Datei löschen.
;
::err    pha                                ;Fehlerstatus zwischenspeichern.

        jsr    CloseRecordFile      ;VLIR-Datei schließen.
        LoadW  r0,paintFileName
        jsr    DeleteFile           ;Beschädigte Datei löschen.

        pla
        tax                                ;Fehlerstatus zurücksetzen.
::1      plp                                ;Interrupt-Status zurücksetzen.
        rts

;
; Neues GeoPaint-Dokument erstellen.
;
:paintCreateFile

        jsr     :delete              ;Vorhandene Datei löschen.

        LoadW  r9 ,HdrGP_Dok
        LoadB   r10L,$00
        jsr     SaveFile             ;Leeres Dokument speichern.
        txa                                ;Diskettenfehler?
        bne     :delete              ; => Ja, Abbruch...

```

```

LoadW r0,paintFileName
jsr   OpenRecordFile    ;Neues Dokument öffnen.
txa   ;Diskettenfehler?
bne   :delete           ; => Ja, Abbruch...

::loop
lda   #0
pha
jsr   AppendRecord      ;Datensatz einfügen.
pla
cpx   #$00              ;Diskettenfehler?
bne   :delete           ; => Ja, Abbruch...
clc
adc   #$01
cmp   #45               ;45 Datensätze entsprechen
bcc   :loop             ;90 Cards Bildgröße.

jsr   UpdateRecordFile  ;VLIR-Datei aktualisieren.
txa   ;Diskettenfehler?
bne   :delete           ; => Ja, Abbruch...

jsr   CloseRecordFile   ;GeoPaint-Dokument schließen.
txa   ;Diskettenfehler?
beq   :done             ; => Nein, Ende...

::delete
LoadW r0,paintFileName
jsr   DeleteFile        ;Vorhandenes Photoscrap löschen.
txa   ;Diskettenfehler?
beq   :done             ; => Nein, Ende...

::done
rts

;
; Grafikdaten in Datei schreiben.
;
; Dabei werden die Grafikdaten von zwei
; Card-Zeilen (2x8=16 Pixel Höhe) und
; die dazugehörigen Farbdaten in einem
; VLIR-Datensatz gespeichert.
;
:paintWriteFile

LoadW a0,SCREEN_BASE    ;Startadresse Grafikdaten.
LoadW a2,COLOR_MATRIX   ;Startadresse Farbdaten.

lda   #$00              ;Zeiger auf ersten Datensatz.
jsr   PointRecord

lda   #00               ;Zeiger auf erste Grafik-Zeile.
sta   r12H

::loop

jsr   paintCopyData     ;Bildschirmdaten einlesen.
; (2*640 Grafik, 2*80 Farbe).
jsr   paintPackData     ;Bildschirmdaten packen.

```

```

        LoadW  r7,dataPacked      ;Zeiger auf Zwischenspeicher.
        jsr    WriteRecord        ;Datensatz auf Diskette schreiben.
        txa
        bne    :err

        jsr    NextRecord         ;Zeiger auf nächsten Datensatz.
        txa
        bne    :err

        inc     r12H               ;Zähler korrigieren.
        lda     r12H               ;13x2 Cards = max.26 Cards Höhe.
        cmp     #13                ;Alle Daten kopiert?
        bcc     :loop              ; => Nein, weiter...
::err    rts

;
;  Daten in Zwischenspeicher kopieren.
;
;  Die Daten werden aus dem Bildschirm-
;  speicher zuerst ungepackt in den
;  Zwischenspeicher kopiert:
;    320 Byte (Grafik-Zeile #1) + 320 Leerbytes
;  + 320 Byte (Grafik-Zeile #2) + 320 Leerbytes
;  +   8 Byte (reserviert)
;  +  40 Byte (Farben-Zeile #1) +  40 Leerbytes
;  +  40 Byte (Farben-Zeile #2) +  40 Leerbytes
;
; Übergabe:
; a0 = Zeiger auf Grafikdaten.
; a2 = Zeiger auf Farbdaten.
;
:paintCopyData jsr    i_FillRam      ;Zwischenspeicher für Grafikdaten
               w      1280 +8        ;löschen (incl. 8 Füllbytes).
               w      dataUnpacked + 0
               b      $00

               jsr    i_FillRam      ;Zwischenspeicher für Farbdaten
               w      160            ;mit Vorgabewert füllen.
               w      dataUnpacked +1288
               b      $bf

               lda     #< dataUnpacked
               sta     a1L
               lda     #> dataUnpacked
               sta     a1H            ;Zeiger auf ungepackte Grafikdaten.

               lda     #< dataUnpacked +1288
               sta     a3L
               lda     #> dataUnpacked +1288
               sta     a3H            ;Zeiger auf ungepackte Farbdaten.

               lda     r12H
               cmp     #12            ;Letzte Doppelzeile schreiben?
               beq     :skip          ; => Ja, nur eine Zeile kopieren.

```

```

        jsr    getDataGrfx      ;Grafikdaten in Zwischenspeicher.
        jsr    getDataCols      ;Farbdaten in Zwischenspeicher.

::skip    jsr    getDataGrfx      ;Grafikdaten in Zwischenspeicher.
        jmp    getDataCols      ;Farbdaten in Zwischenspeicher.

;
; Grafikdaten einlesen.
;
; Dabei werden die Grafikdaten aus dem
; Bildschirmspeicher in den Puffer für
; die ungepackten Daten kopiert.
;
;
getDataGrfx    MoveW    a0 ,r0          ;320 Grafikdaten kopieren.
               MoveW    a1 ,r1
               LoadW    r2 ,320
               jsr    MoveData

               AddVW    320,a0          ;Zeiger auf nächste Grafikzeile.
               AddVW    640,a1          ;Zeiger auf Speicher korrigieren.
               rts

;
; Farbdaten einlesen.
;
; Dabei werden die Farbdaten aus dem
; Farbspeicher in den Puffer für die
; ungepackten Daten kopiert.
;
;
getDataCols    MoveW    a2 ,r0          ;40 Farbdaten kopieren.
               MoveW    a3 ,r1
               LoadW    r2 ,40
               jsr    MoveData

               AddVW    40 ,a2          ;Zeiger auf nächste Farbzeile.
               AddVW    80 ,a3          ;Zeiger auf Speicher korrigieren.
               rts

;
; Daten packen.
;
; dataUnpacked = Ungepackte Daten.
; dataPacked   = Zwischenspeicher.
;
;
; Verwendete Register:
; a0 = Zeiger auf Zwischenspeicher für Grafikdaten.
; a1 = Zeiger auf VLIR-Speicher für Grafikdaten.
; a2 = Zeiger auf Zwischenspeicher für Farbdaten.
; a3 = Zeiger auf VLIR-Speicher für Farbdaten.
; a6 = Anzahl der noch zu bearbeitenden Bytes.
; a7 = Zwischenspeicher 8-Byte-Blocks.
; a8L = Anzahl 8-Byte-Blocks.
; a8H = Anzahl identische 8-Byte-Blocks.
; a9L = Anzahl identische Bytes.
; a9H = Anzahl ungepackter Bytes.

```



```

;
; Rückgabe:
; r2 = Anzahl gepackte Datenbytes.
;
;
; Max. Anzahl Datenbyte berechnen:
;
; 80 Cards Grafik x 8 Byte x 2 Zeilen
; + 8 Füllbyte
; + 80 Cards Farbe x 2 Zeilen
; = 1448 Bytes
;
; Bytes aus Zwischenspeicher einlesen,
; packen und in Speicher für Geopaint-
; Datensatz kopieren.
:paintPackData LoadW r0,dataUnpacked ;Zeiger auf ungepackte Daten.
               LoadW r1,dataPacked   ;Zeiger auf VLIR-Speicher.
               LoadW a6,1448          ;Max. Anzahl Datenbyte.

               lda    #$00
               sta    a9L              ;Anzahl identische Einzelbytes.
               sta    a8H              ;Anzahl identische 8-Byte-Blocks.

::next         jsr    paintEqualBytes ;Nach gleichen Bytes suchen.
               cmp    #8              ;Mehr als 8 gleiche Bytes?
               bcs    :single         ; => Ja, weiter...

;
; Mehrere 8-Byte-Blöcke?
;
::multi        jsr    paintGet8Block  ;Gleichen 8-Byte-Blöcke suchen.
               cmp    #2              ;Mehr als 1 gleicher Block?
               bcc    :single         ; => Nein, weiter...

               ldy    #< paintPack8Block
               ldx    #> paintPack8Block
               bne    :exec           ;$4x = 8-Byte-Blöcke packen.

;
; Einzelbytes packen oder
; ungepackte Daten?
;
::single       lda    a9L              ;Anzahl zu packender Bytes.
               cmp    #4              ;Mehr als vier Bytes?
               bcc    :unpacked       ; => Ja, Daten nicht packen.

               ldy    #< paintPackSingle
               ldx    #> paintPackSingle
               bne    :exec           ;$8x = Einzelbytes packen.

::unpacked     ldy    #< paintPackNoData
               ldx    #> paintPackNoData
;               bne    :exec           ;$0x = Einzelbytes packen.

```

```

;
; Daten in Zwischenspeicher kopieren,
; gepackt oder ungepackt.
::exec      tya
            jsr    CallRoutine      ;Packroutine aufrufen.

            lda    a6L
            ora    a6H              ;Alle Bytes gepackt?
            bne    :next            ; => Nein, weiter...

            lda    #$00             ;Abschluss-Byte.
            tay    (r1L),y          ; => Farb- und Grafikdaten müssen
            sta    r1L              ; mit einem NULL-Byte enden!
            inc    r1L
            bne    :done
            inc    r1H

::done      lda    r1L              ;Anzahl gepackte Bytes berechnen.
            sec
            sbc    #< dataPacked
            sta    r2L
            lda    r1H
            sbc    #> dataPacked
            sta    r2H

            rts

;
; Identische Datenbyte suchen.
;
; Sucht in den ungepackten Datenbyte
; mehrere gleiche, aufeinanderfolgende
; Einzelbyte.
;
; Übergabe:
; r0 = Zeiger auf ungepackte Daten.
; a6 = Anzahl noch zu packender Daten.
; a9L = Anzahl identische Datenbyte.
;
:paintEqualBytes
            lda    a9L              ;Sind noch gleiche Einzelbytes
            bne    :skip            ;im Speicher? Nein, Daten noch
                                   ;nicht komplett gepackt, nächste
                                   ;Einzelbytes packen.

            ldy    #$00             ;Zeiger auf aktuelles Byte.
            lda    (r0L),y          ;Aktuelles Byte einlesen.
            iny
                                   ;Zeiger auf nächstes Byte.
::loop      cmp    (r0L),y          ;Byte identisch mit aktuellem Byte?
            bne    :exit            ; => Nein, weiter...
            iny                    ;Zähler für gleiche Byte erhöhen.
            cpy    #63              ;Max. 63 gleiche Bytes erreicht?
            bcc    :loop            ; => Nein, weiter...

```

```

::exit      lda    a6H          ;Anzahl gleiche Bytes mit Anzahl
           bne    :1          ;der noch zu packenden Bytes
           cpy    a6L          ;vergleichen.
           bcc    :1
           beq    :1
           ldy    a6L          ;Anzahl Bytes auf Restbytes setzen.
:::1        tya    a6L          ;Anzahl gleicher Einzelbytes
           sta    a9L          ;zwischenspeichern.

::skip      rts

;
; Identische 8-Byte-Blöcke suchen.
;
; Sucht in den ungepackten Datenbyte
; mehrere gleiche, aufeinanderfolgende
; 8-Byte-Blöcke.
;
; Übergabe:
; r0 = Zeiger auf ungepackte Daten.
; a6 = Anzahl noch zu packender Daten.
; a8H = Anzahl 8-Byte-Blöcke.
;
; Rückgabe:
; AKKU = Anzahl 8-Byte-Blöcke.
;
; paintGet8Block
           lda    a8H          ;Sind noch gleiche 8-Byte-Blöcke
           bne    :skip        ;im Speicher? Nein, Daten noch
                               ;nicht komplett gepackt, nächsten
                               ;8-Byte-Block packen.

           lda    a6L          ;Anzahl noch zu packender Bytes
           sta    a8L          ;einlesen und durch 8 teilen.
           lda    a6H          ;Dadurch noch verbleibende 8-Byte-
           lsr     a8L          ;Blöcke berechnen.
           ror     a8L
           lsr     a8L
           ror     a8L
           lsr     a8L
           ror     a8L

           lda    a8L          ;Mehr als 2x 8-Byte-Block übrig?
           cmp    #2          ; => Ja, weiter...
           bcs    :test

::skip      lda    #$00        ;Packen nicht effektiv, da zu
           rts                ;wenig Bytes zum packen übrig.

::test      cmp    #63 +1      ;Mehr als 63x 8-Byte-Block übrig?
           bcc    :init        ; => Weniger als 63, weiter...
           lda    #63          ;Max.-Wert 63 für 8-Byte-Blöcke
           sta    a8L          ;in einen Packdurchgang setzen.

```

```

::init      lda    r0L          ;Zeiger auf den ersten 8-Byte-
            sta    a7L          ;Block setzen.
            lda    r0H
            sta    a7H

            ldx    #1           ;Zähler initialisieren.
::next      lda    a7L          ;Zeiger auf nächsten 8-Byte-Block.
            clc
            adc    #8
            sta    a7L
            bcc    :1
            inc    a7H

::1         ldy    #8 -1
::loop      lda    (r0L),y      ;Bytes in nächstem 8-Byte-Block
            cmp    (a7L),y      ;gleich wie aktueller 8-Byte-Block?
            bne    :exit        ; => Nein, Ende...
            dey
            bpl    :loop        ; => Ja, nächstes Byte testen...

            inx                ;Max. Wert für gleiche Blöcke
            cpx    a8L          ;erreicht (max. 63)?
            bcc    :next        ; => Nein, weiter...

::exit      txa
            sta    a8H          ;Anzahl gleicher 8-Byte-Blocks
            rts                ;zwischenspeichern.

;
; Daten packen ($4x)
;
; Packt mehrere 8-Byte-Blöcke. Die
; 8-Byte sind nicht gepackt.
;
; Übergabe:
; r0 = Zeiger auf ungepackte Daten.
; r1 = Zeiger auf Zwischenspeicher.
; a6 = Anzahl noch zu packender Daten.
; a8H = Anzahl 8-Byte-Blöcke.
;
;paintPack8Block
            lda    a8H          ;Anzahl 8-Byte-Blocks einlesen.
            ora    #$40         ;Kompressions-Flag setzen.

            ldy    #$00         ;Zeiger auf Zwischenspeicher.
            sta    (r1L),y      ;Kompressionsbyte setzen.

            inc    r1L          ;Zeiger auf nächstes Byte im
            bne    :1           ;Zwischenspeicher.
            inc    r1H

::1         ldy    #$07         ;8-Byte-Block in VLIR-Speicher
            jsr    paintCopyYRegByt ;übertragen.

```

```

        lda    a8H                ;Anzahl 8-Byte-Blocks einlesen und
        sta    a7L                ;in Einzelbytes umrechnen.
        lda    #$00
        asl    a7L
        rol
        asl    a7L
        rol
        asl    a7L
        rol
        sta    a7H

        lda    a7L                ;Zeiger auf Grafikdaten um Anzahl
        clc                                ;gepackter 8-Byte-Blocks erhöhen.
        adc    r0L
        sta    r0L
        lda    a7H
        adc    r0H
        sta    r0H

        lda    r1L                ;Zeiger für Zwischenspeicher auf
        clc                                ;nächstes Byte setzen.
        adc    #8
        sta    r1L
        bcc    :2
        inc    r1H

::2      lda    a6L                ;Anzahl noch zu packender Bytes
        sec                                ;korrigieren.
        sbc    a7L                ; => In :a7 steht die Anzahl der
        sta    a6L                ; gepackten 8-Byte-Blöcke, umge-
        lda    a6H                ; rechnet in Einzelbytes.
        sbc    a7H
        sta    a6H

        jmp    paintClearFlags    ;8-Byte/Einzelbyte-Flag löschen.

;
;  Daten packen ($8x)
;
;  Packt mehrere identische Einzelbyte.
;
;  Übergabe:
;  r0 = Zeiger auf ungepackte Daten.
;  r1 = Zeiger auf Zwischenspeicher.
;  a6 = Anzahl noch zu packender Daten.
;  a9L = Anzahl Einzelbytes.
;
:paintPackSingle
        lda    a9L                ;Anzahl Einzelbytes einlesen.
        ora    #$80                ;Kompressions-Flag setzen.

        ldy    #$00                ;Zeiger auf VLIR-Speicher.
        sta    (r1L),y            ;Kompressionsbyte setzen.

```

```

        lda    (r0L),y      ;Zu packendes Byte einlesen und
        iny
        sta    (r1L),y      ;als Packbyte in den Zwischen-
                               ;speicher schreiben.

        lda    r1L          ;Zeiger für Zwischenspeicher auf
        clc                ;nächstes Byte setzen.
        adc    #2
        sta    r1L
        bcc    :1
        inc    r1H

::1      lda    a9L          ;Zeiger auf Grafikdaten um Anzahl
        clc                ;gepackter Einzelbytes erhöhen.
        adc    r0L
        sta    r0L
        bcc    :2
        inc    r0H

::2      lda    a6L          ;Anzahl noch zu packender Bytes
        sec                ;korrigieren.
        sbc    a9L          ; => In :a9L steht die Anzahl der
        sta    a6L          ; gepackten Einzelbytes.
        bcs    paintClearFlags
        dec    a6H

;*** Flags für 8-Byte-Blöcke/Einzelbytes löschen.
:paintClearFlags
        lda    #$00
        sta    a9L          ;Anzahl Einzelbyte löschen.
        sta    a8H          ;Anzahl 8-Byte-Blocks löschen.
        rts

;
; Daten ungepackt speichern ($01-$3f)
;
; Die Daten werden ungepackt in den
; Zwischenspeicher kopiert.
;
; Übergabe:
; r0 = Zeiger auf ungepackte Daten.
; r1 = Zeiger auf Zwischenspeicher.
; a6 = Anzahl noch zu packender Daten.
; a9H = Anzahl ungepackte Bytes.
;
:paintPackNoData
        jsr    paintCountBytes ;Ungepackte Bytes zählen.

        lda    a9H          ;Anzahl ungepackter Bytes.
        ldy    #$00         ;Zeiger auf VLIR-Speicher.
        sta    (r1L),y      ;Kompressionsbyte setzen.

        inc    r1L          ;Zeiger auf nächstes Byte im
        bne    :1           ;Zwischenspeicher.
        inc    r1H

```

```

::1      ldy    a9H          ;Anzahl ungepackter Bytes in
        dey    ;Zwischenspeicher kopieren.
        jsr    paintCopyYRegByt

        lda    a9H          ;Zeiger auf Grafikdaten um Anzahl
        clc    ;ungepackter Bytes erhöhen.
        adc    r0L
        sta    r0L
        bcc    :2
        inc    r0H

::2      lda    a9H          ;Zeiger für VLIR-Speicher auf
        clc    ;nächstes Byte setzen.
        adc    r1L
        sta    r1L
        bcc    :3
        inc    r1H

::3      lda    a6L          ;Anzahl noch zu packender Bytes
        sec    ;korrigieren.
        sbc    a9H          ; => In :a9H steht die Anzahl der
        sta    a6L          ;   ungepackten Einzelbytes.
        bcs    :4
        dec    a6H

::4      rts

;
; Anzahl Bytes aus Grafikspeicher in
; Zwischenspeicher kopieren.
;
; Übergabe:
; yReg = Anzahl Bytes -1,
;       max. 128 Bytes!
:paintCopyYRegByt  lda      (r0L),y      ;Byte einlesen und in
                   sta      (r1L),y      ;Zwischenspeicher kopieren.
                   dey      ;Alle Bytes kopiert?
                   bpl      paintCopyYRegByt ; => Nein, weiter...
                   rts

;
; Anzahl ungepackte Daten berechnen.
;
; Übergabe:
; r0 = Zeiger auf ungepackte Daten.
; a6 = Anzahl noch zu packender Daten.
;
; Rückgabe:
; a9H = Anzahl ungepackte Daten.
;
:paintCountBytes  lda      #$01          ;Max. Anzahl ungepackter Bytes auf
                   sta      a9H          ;Startwert setzen.

```

```

        PushW  r0           ;Zeiger auf Grafikdaten retten.
        PushW  a6           ;Anzahl zu packender Bytes retten.
        jsr    paintPosNxByte ;Zeiger auf nächstes Byte setzen.
:::1      lda    a6L         ;Weitere Bytes in Grafikspeicher
        ora    a6H         ;zum packen vorhanden?
        beq    :exit        ; => Nein, Ende...

        jsr    paintClearFlags ;8-Byte/Einzelbyte-Flags löschen.

        jsr    paintEqualBytes ;Gleiche Einzelbytes suchen.
        cmp    #4           ;Mehr als vier gleiche Bytes?
        bcs    :exit        ; => Ja, Abbruch. Ab hier ist das
        ;packen über Anzahl gleicher Bytes
        ;wieder effektiver !!!

        jsr    paintGet8Block ;Gleiche 8-Byte-Blocks suchen.
        cmp    #2           ;Mehr als zwei 8-Byte-Blocks?
        bcs    :exit        ; => Ja, Abbruch. Ab hier ist das
        ;packen über die Anzahl gleicher
        ;8-Byte-Blocks wieder effektiver!

        jsr    paintPosNxByte ;Zeiger auf nächstes Byte.

        inc    a9H         ;Anzahl ungepackter Bytes +1.
        lda    a9H
        cmp    #63         ;Max. 63 Bytes gefunden?
        bcc    :1          ;Nein weiter...

        jsr    paintClearFlags ;8-Byte/Einzelbyte-Flags löschen.
:::exit   PopW   a6         ;Anzahl noch zu packender Bytes
        PopW   r0         ;und Zeiger auf Grafikdaten wieder
        rts             ;zurücksetzen.

;
; Zeiger auf ungepackte Daten und
; Anzahl noch zu packender Daten
; korrigieren.
;
:paintPosNxByte
        inc    r0L         ;Zeiger auf nächstes Byte der
        bne    :1          ;Grafikdaten setzen.
        inc    r0H

:::1      lda    a6L         ;Anzahl noch zu packender Bytes
        bne    :2          ;korrigieren.
        dec    a6H
:::2      dec    a6L


        rts

```



```

;
; Variablen.
;
:paintFileName b "Screen Capture",NULL

;
; Header für Geopaint-Datei.
;
:HdrGP_Dok      w paintFileName
                b $03,$15
                j
                

:HdrGP_068      b $83
:HdrGP_069      b APPL_DATA
:HdrGP_070      b VLIR
:HdrGP_071      w $0000,$ffff,$0000
:HdrGP_077      b "Paint Image "           ;Klasse.
:HdrGP_089      b "V1.1"                   ;Version.
:HdrGP_093      b $00                       ;NULL-Byte.
:HdrGP_094      b $00,$00,$00              ;Reserviert.
:HdrGP_097      b NULL                     ;Autor.
:HdrGP_098      e HdrGP_097 +20             ;Reserviert.
:HdrGP_117      b "geoPaint "               ;Application.
:HdrGP_129      b "V2.0"                   ;Version.
:HdrGP_133      b $00                       ;NULL-Byte.
:HdrGP_134      b $01                       ;Flag für "Farbe an".
:HdrGP_135      s 25                       ;Reserviert.
:HdrGP_160      b NULL

```

Zum Abschluss noch ein Screenshot:

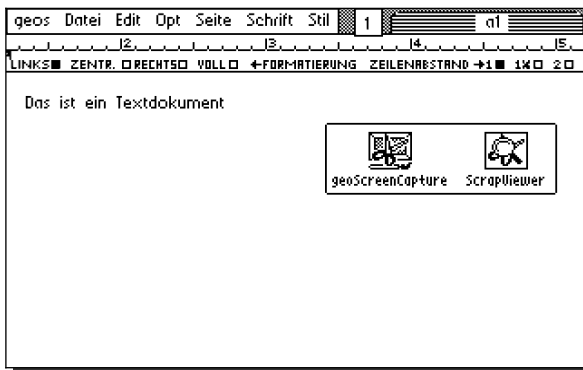


Bild L4.1: geoScreenCapture

Hier wurde das Programm aus GeoWrite als DeskAccessory gestartet. Nach dem Start erscheint ein Auswahlrahmen, der mit den Cursor-Tasten um die beiden Icon platziert wurde. Über die Taste **[RETURN]** wird das Photoscrap erstellt.

M.5 Demo/Application: "ScrapViewer"

Mit dem Programm "geoScreenCapture" lassen sich auch größere Photoscrap erstellen, die man nicht mehr in GeoWrite einbinden kann bzw. die man in GeoPaint nur skaliert einfügen kann. Um zu testen ob diese Photoscraps korrekt erstellt wurden, und als zusätzliches Beispiel zur GEOS-Routine *BitmapClip*, folgt hier das Programm "ScrapViewer".

Mit Hilfe des Programms wird das Photoscrap auf dem eingestellten Laufwerk geöffnet und am Bildschirm angezeigt. Ist das Photoscrap größer als der Bereich der Dialogbox, dann kann der Ausschnitt über die Cursor-Tasten verschoben werden.

Leider sieht man dabei auch einen großen Nachteil der GEOS-Routinen: Diese sind zwar sehr flexibel, aber auch sehr langsam. Beim verschieben des Bildausschnitts wird der Hauptteil der Grafik über die GEOS-Routine *MoveData* verschoben, in dem direkt auf den Grafikspeicher zugegriffen wird. Es wird daher nur der neu Teil der Grafik über *BitmapClip* ausgegeben, und das deutlich langsamer obwohl viel weniger Daten angezeigt werden müssen als zuvor verschoben wurden.

Die Hauptaufgabe ist aber eine andere, nämlich nur ein Photoscrap am Bildschirm anzuzeigen. Über die Tasten [A], [B], [C] und [D] kann das Laufwerk gewechselt werden. Ist auf dem neuen Laufwerk kein Photoscrap vorhanden, dann wird eine entsprechende Meldung angezeigt.

Hier der Quelltext zu "ScrapViewer":

```
;
; Symboltabellen einbinden.
;
if .p
    t    "TopSym"
    t    "TopMac"
endif

;
; GEOS-Header definieren.
;
n    "ScrapViewer"
c    "ScrapViewer V1.0",NULL
a    "Markus Kanet",NULL

f    APPLICATION
z    $80 ;Nur GEOS64.

o    APP_RAM

h    "A simple Photoscrap viewer"
h    "use Cursor keys and a,b,c,d to select drive."

i
```



```

;
; Fensterdaten definieren:
;
; Linke, obere Ecke Dialogbox:
:WINPOS_X    = $05 ;Cards
:WINPOS_Y    = $18 ;Pixel
;
; Breite/Höhe für Dialogbox:
:WINDOW_X    = $1d ;Cards
:WINDOW_Y    = $70 ;Pixel
;
; Offset für Scrap-Anzeige:
:OFFSET_X    = $01 ;Cards
:OFFSET_Y    = $08 ;Pixel
;
; max. Breite/Höhe für Scrap-Anzeige:
:CLIP_X      = WINDOW_X -OFFSET_X*2
:CLIP_Y      = WINDOW_Y -OFFSET_Y*2
;
; Ladeadresse für Photoscrap:
:SCRAP_BASE  = $1000
:SCRAP_SIZE  = $7000
;
; Startadresse Photoscrap_Daten:
:SCRAP_DATA  = SCRAP_BASE +3

; Größe Photoscrap.
:scrapWidth  = SCRAP_BASE +0
:scrapHeight = SCRAP_BASE +1

;
; Photoscrap-Viewer
;
:Start       LoadW r0,dbox           ;Zeiger auf Dialogbox-Daten.
            jsr   DoDlgBox          ;Dialogbox öffnen.
::exit       jmp   EnterDeskTop      ;Zurück zum DeskTop.

;
; Anzeigebereich Photoscrap löschen.
;
:ClearClip   lda   #2
            jsr   SetPattern         ;Füllmuster setzen.

; Linke, obere Ecke definieren.
            LoadB r2L,(WINPOS_Y +OFFSET_Y)
            LoadW r3 ,(WINPOS_X +OFFSET_X) *8

; Rechte, untere Ecke definieren.
            LoadB r2H,(WINPOS_Y +OFFSET_Y +CLIP_Y) -1
            LoadW r4 ,(WINPOS_X +OFFSET_X +CLIP_X) *8 -1

; Anzeigebereich löschen.
            jsr   Rectangle
            rts

```

```

;
; Photoscrap einlesen
;
:loadPScrap
    LoadW    r6,ScrapName    ;Zeiger auf Dateiname.
    jsr      FindFile        ;Photoscrap suchen.
    txa      ;Diskettenfehler?
    bne      :err            ; => Ja, Abbruch...

if FALSE
    jsr      i_FillRam        ;Debug-Modus:
    w        $7000           ;Speicher löschen.
    w        $1000
    b        $bd
endif

    lda      dirEntryBuf +1   ;Zeiger auf ersten Track/Sektor
    sta      r1L              ;des Photoscrap einlesen.
    lda      dirEntryBuf +2
    sta      r1H

    LoadW    r7,SCRAP_BASE   ;Ladeadresse definieren.
    LoadW    r2,SCRAP_SIZE   ;Max. Puffergröße festlegen.
    jsr      ReadFile         ;Photoscrap einlesen.
    txa      ;Diskettenfehler?
    beq      :done            ; => Nein, Ende...

::err      ldx      #$ff      ;Flag: "Kein Scrap im Speicher"
::done     stx      Flag_ScrapOK ;Scrap-Status speichern.

    rts

;
; Dialogbox-Menü initialisieren
;
:InitDBoxMenu
    lda      #ST_WR_FORE     ;Nur in Vordergrund schreiben.
    sta      dispBufferOn

    lda      #< keyDBoxMenu   ;Tastaturabfrage installieren.
    sta      keyVector +0
    lda      #> keyDBoxMenu
    sta      keyVector +1

;
; Anzeige für Photoscrap initialisieren
;
:DrawFirstClip
    jsr      loadPScrap       ;Photoscrap-Datei laden.

    bit      Flag_ScrapOK     ;Photoscrap im Speicher?
    bmi      ScrapError       ; => Fehlermeldung ausgeben.

::ok       jmp      InitPhotoScrap

```

```

;
; Kein Photoscrap vorhanden
; Fehlermeldung anzeigen
;
:ScrapError    LoadW  r11,(WINPOS_X +OFFSET_X) *8 +16
               LoadB  r1H, WINPOS_Y +OFFSET_Y      +16

               LoadW  r0,Text_NoData
               jsr     PutString

               rts

;
; Photoscrap initialisieren
;
:InitPhotoScrap
               lda     #$00                ;Offset für Scrap-Anzeige
               sta     clipXPos            ;initialisieren.
               sta     clipYPos +0
               sta     clipYPos +1

; Koordinate für Scrap-Ausgabe:
; - x-Koordinate in Cards.
; - y-Koordinate in Pixel.
               LoadB  r1L,WINPOS_X +OFFSET_X
               LoadB  r1H,WINPOS_Y +OFFSET_Y

; Größe für Scrap-Ausgabe:
; - x-Größe in Cards.
; - y-Größe in Pixel.
               LoadB  r2L,CLIP_X
               LoadB  r2H,CLIP_Y

; Offset für Scrap-Ausgabe:
; - x-Offset in Cards.
; - y-Offset in Pixel.
               MoveB   clipXPos,r11L
:DrawCurClip  MoveW   clipYPos,r12

;
; Ausschnitt Photoscrap anzeigen
;
:DrawCurYClip LoadW  r0,SCRAP_DATA        ;Zeiger auf Scrap-Daten.

               lda     scrapWidth
               sec
               sbc     r11L                ;Anzahl Cards am Anfang überlesen.
               bcc     :small_x

;
               sec
               sbc     r2L
               sta     r11H                ;Anzahl Cards am Ende überlesen.
               bcs     :print

```

```

;
; Photoscrap ist schmaler als Anzeige
;
;
::small_x      lda    #0
               sta    r11L           ;Keine Cards am Anfang überlesen.
               sta    r11H           ;Keine Cards am Ende überlesen.

               lda    scrapWidth     ;Max. Breite Photoscrap setzen.
               sta    r2L

;
; Aktuellen Ausschnitt anzeigen.
;
;
::print         jsr    defClipSize    ;Höhe Photoscrap testen.
               jmp    BitmapClip     ;Ausschnitt anzeigen.

;
; Höhe Photoscrap-Ausschnitt prüfen
;
;
::defClipSize   lda    scrapHeight +1 ;Höhe > 256 Pixel?
               bne    :1             ; => Ja, kein Test erforderlich.

               lda    scrapHeight +0 ;Höhe > 200 Pixel?
               cmp    r2H
               bcs    :1             ; => Ja, weiter...
               sta    r2H           ;Max. Höhe festlegen.

::1             lda    scrapWidth     ;Breite > 40 Cards?
               cmp    r2L
               bcs    :2             ; => Ja, weiter...
               sta    r2L           ;Max. Breite festlegen.
::2             rts                 ;Ende.

;
; Tastaturabfrage
;
;
::keyDBoxMenu   lda    keyData        ;Tastencode einlesen.
               ldx    #0             ;Zeiger auf Anfang Tastentabelle.
::1             cmp    tabKeyData,x   ;Tastencode gefunden?
               beq    execKey        ; => Ja, ausführen.
               inx
               cpx    #MAX_KEYS      ;Alle Tasten geprüft?
               bcc    :1             ; => Nein, weiter...
::exit          rts                 ;Zurück zur Mainloop.

;
; Tastenroutine ausführen
;
;
::execKey       txa
               asl
               tay
               lda    tabKeyAdr +0,y  ;Adresse für Tastenroutine
               ldx    tabKeyAdr +1,y  ;einlesen und ausführen.
               jmp    CallRoutine

```

```

;
; Tastencodes für Menüfunktionen
;
;
:tabKeyData    b $1e ;right, MoveLeft
               b $08 ;left , MoveRight
               b $11 ;down , MoveDown
               b $10 ;up  , MoveUp
               b $61 ;a   , Laufwerk A:
               b $62 ;b   , Laufwerk B:
               b $63 ;c   , Laufwerk C:
               b $64 ;d   , Laufwerk D:
               b $78 ;x   , Desktop
               b $0d ;RET , Desktop

:tabKeyDataEnd
:MAX_KEYS      = tabKeyDataEnd - tabKeyData

;
; Adressen der Menüroutinen.
;
:tabKeyAdr      w MoveLeft
               w MoveRight
               w MoveDown
               w MoveUp
               w OpenDriveA
               w OpenDriveB
               w OpenDriveC
               w OpenDriveD
               w EnterDeskTop
               w EnterDeskTop

;
; Tastenmenü: Laufwerk wechseln
;
:OpenDriveA     ldx    #8                ;Laufwerk A:
               b $2c
:OpenDriveB     ldx    #9                ;Laufwerk B:
               b $2c
:OpenDriveC     ldx    #10               ;Laufwerk C:
               b $2c
:OpenDriveD     ldx    #11               ;Laufwerk D:
               lda     driveType -8,x    ;Ist Laufwerk verfügbar?
               beq     :exit              ; => Nein, Abbruch...
               txa
               jsr     SetDevice          ;Laufwerk aktivieren.

               lda     #$ff
               sta     Flag_ScrapOK      ;Scrap-Status löschen.
               jsr     ClearClip         ;Anzeigebereich löschen.

               jsr     OpenDisk           ;Diskette öffnen.
               txa
               bne     :exit              ;Diskettenfehler?
               ; => Ja, Abbruch...

```

```

        jsr    DrawFirstClip    ;Photoscrap anzeigen.
::exit    rts                    ;Ende.
;
; Ausschnitt nach links schieben
;
:leftCol0 = SCREEN_BASE +(WINPOS_Y +OFFSET_Y)*40 +(WINPOS_X +OFFSET_X +1)
*8
:leftCol1 = leftCol0 -8
:MoveLeft    bit    Flag_ScrapOK    ;Photoscrap im Speicher?
             bmi    :exit            ; => Nein, Ende...

             lda    clipXPos         ;Ausschnitt bereits am
             clc                     ;rechten Rand?
             adc    #CLIP_X
             bcs    :exit
             cmp    scrapWidth
             bcc    :ok              ; => Nein, verschieben...

::exit    rts

;
; Ausschnitt verschieben
;
:ok        inc    clipXPos           ;Offset für Anzeige ändern.

             LoadW  r0,leftCol0      ;Spalte #1
             LoadW  r1,leftCol1      ;Spalte #0

             ldx    #CLIP_X          ;Breite des Ausschnitts
             dex                     ;von Cards nach Pixel wandeln.
             txa                     ;Achtung: Max. 32 Cards!
             asl
             asl
             sta    r2L
             lda    #$00
             sta    r2H

::loop     lda    #(CLIP_Y/8)        ;Zeilenzähler initialisieren.
             pha
             jsr    MoveData          ;Daten verschieben.
             AddVW  40*8,r0          ;Zeiger auf nächste Zeile.
             AddVW  40*8,r1          ;Zeiger auf nächste Zeile.
             pla
             sec
             sbc    #$01              ;Alle Zeilen verschoben?
             bne    :loop            ; => Nein, weiter...

; x-Koordinate setzen.
             lda    #WINPOS_X +OFFSET_X -1
             clc
             adc    #CLIP_X
             sta    r1L

```



```

; y-Koordinate setzen.
    LoadB r1H,WINPOS_Y +OFFSET_Y

; Größe des Ausschnitts definieren.
    LoadB r2L,1           ;Breite in Cards.
    LoadB r2H,CLIP_Y      ;Höhe in Pixel.

; x-Offset berechnen.
    lda clipXPos           ;x-Offset in Cards.
    clc
    adc #CLIP_X
    sta r11L
    dec r11L

; Letzte Spalte ausgeben.
    jsr DrawCurClip       ;Daten über BitmapClip ausgeben.

    rts                   ;Ende.

;
; Ausschnitt nach rechts schieben
;
:rightCol0 = SCREEN_BASE +(WINPOS_Y +OFFSET_Y)*40 +(WINPOS_X +OFFSET_X) *8
:rightCol1 = rightCol0 +8
:MoveRight
    bit Flag_ScrapOK       ;Photoscrap im Speicher?
    bmi :exit              ; => Nein, Ende...

    ldx clipXPos           ;Bereits am linken Rand?
    bne :ok                ; => Nein, verschieben...

::exit    rts

;
; Ausschnitt verschieben
;
::ok      dex
          stx clipXPos      ;Offset für Anzeige ändern.

          LoadW r0,rightCol0 ;Spalte #x -1
          LoadW r1,rightCol1 ;Spalte #x

          ldx #CLIP_X       ;Breite des Ausschnitts
          dex               ;von Cards nach Pixel wandeln.
          txa               ;Achtung: Max. 32 Cards!
          asl
          asl
          asl
          sta r2L
          lda #$00
          sta r2H

          lda #(CLIP_Y/8)   ;Zeilenzähler initialisieren.
::loop    pha

```

```

        jsr    MoveData          ;Daten verschieben.
        AddVW  40*8,r0           ;Zeiger auf nächste Zeile.
        AddVW  40*8,r1           ;Zeiger auf nächste Zeile.
        pla
        sec
        sbc    #$01              ;Alle Zeilen verschoben?
        bne    :loop             ; => Nein, weiter...

; x-/y-Koordinate setzen.
        LoadB  r1L,WINPOS_X +OFFSET_X
        LoadB  r1H,WINPOS_Y +OFFSET_Y

; Größe des Ausschnitts definieren.
        LoadB  r2L,1             ;Breite in Cards.
        LoadB  r2H,CLIP_Y        ;Höhe in Pixel.

; x-Offset setzen.
        MoveB   clipXPos,r11L     ;x-Offset in Cards.

; Erste Spalte ausgeben.
        jsr     DrawCurClip       ;Daten über BitmapClip ausgeben.

        rts                      ;Ende.

;
; Ausschnitt nach unten schieben
;
:downCol0 = SCREEN_BASE +(WINPOS_Y +OFFSET_Y)*40 +(WINPOS_X +OFFSET_X)*8
:downCol1 = downCol0 +40*8
:MoveDown    bit    Flag_ScrapOK    ;Photoscrap im Speicher?
              bmi    :exit           ; => Nein, Ende...

              lda    clipYPos +0
              clc
              adc    #CLIP_Y
              sta    r1L
              lda    clipYPos +1
              adc    #$00
              sta    r1H

              CmpW   r1,scrapHeight  ;Bereits am unteren Rand?
              bcc    :ok             ; => Nein, verschieben...

::exit      rts

;
; Ausschnitt verschieben
;
::ok        lda    clipYPos +0       ;Offset für Anzeige ändern.
              clc
              adc    #8
              sta    clipYPos +0
              bcc    :1
              inc    clipYPos +1

```

```

::1      LoadW  r0,downCol1      ;Zeile #y -1
        LoadW  r1,downCol0      ;Zeile #y

        LoadW  r2,CLIP_X *8      ;Anzahl Bytes in einer Zeile.

::loop   lda    #(CLIP_Y/8) -1    ;Zeilenzähler initialisieren.
        pha
        jsr    MoveData          ;Daten verschieben.
        AddVW  40*8,r0           ;Zeiger auf nächste Zeile.
        AddVW  40*8,r1           ;Zeiger auf nächste Zeile.
        pla
        sec
        sbc    #$01              ;Alle Zeilen verschoben?
        bne    :loop             ; => Nein, weiter...

; x-/y-Koordinate setzen.
        LoadB  r1L,WINPOS_X +OFFSET_X
        LoadB  r1H,WINPOS_Y +OFFSET_Y +CLIP_Y -8

; Größe des Ausschnitts definieren.
        LoadB  r2L,CLIP_X        ;Breite in Cards.
        LoadB  r2H,8              ;Höhe in Pixel.

; x-Offset setzen.
        MoveB   clipXPos,r11L      ;x-Offset in Cards.

; y-Offset setzen.
        lda    clipYPos +0         ;y-Offset in Pixel.
        clc
        adc    #< (CLIP_Y -8)
        sta    r12L
        lda    clipYPos +1
        adc    #> (CLIP_Y -8)
        sta    r12H

; Unterste Spalte ausgeben.
        jsr    DrawCurYClip       ;Daten über BitmapClip ausgeben.

        rts                        ;Ende.

;
; Ausschnitt nach oben schieben
;
upCol = SCREEN_BASE +(WINPOS_Y +OFFSET_Y)*40 +(WINPOS_X +OFFSET_X)*8
upCol0 = upCol +(CLIP_Y/8 -1)*40*8
upCol1 = upCol0 -40*8
:MoveUp  bit    Flag_ScrapOK       ;Photoscrap im Speicher?
        bmi    :exit              ; => Nein, Ende...

        lda    clipYPos +0
        ora    clipYPos +1         ;Bereits am oberen Rand?
        bne    :ok                ; => Nein, verschieben...

::exit   rts

```

```

;
; Ausschnitt verschieben
;
::ok      lda    clipYPos +0      ;Offset für Anzeige ändern.
          sec
          sbc    #< $0008
          sta    clipYPos +0
          bcs    :1
          dec    clipYPos +1

::1       LoadW r0,upCol1        ;Zeile #1
          LoadW r1,upCol0        ;Zeile #0

          LoadW r2,CLIP_X *8     ;Anzahl Bytes in einer Zeile.

          lda    #(CLIP_Y/8) -1  ;Zeilenzähler initialisieren.
          pha
          jsr    MoveData         ;Daten verschieben.
          SubVW  40*8,r0         ;Zeiger auf nächste Zeile.
          SubVW  40*8,r1         ;Zeiger auf nächste Zeile.
          pla
          sec
          sbc    #$01             ;Alle Zeilen verschoben?
          bne    :loop           ; => Nein, weiter...

; x-/y-Koordinate setzen.
          LoadB r1L,WINPOS_X +OFFSET_X
          LoadB r1H,WINPOS_Y +OFFSET_Y

; Größe des Ausschnitts definieren.
          LoadB r2L,CLIP_X       ;Breite in Cards.
          LoadB r2H,8            ;Höhe in Pixel.

; x-Offset setzen.
          MoveB  clipXPos,r11L    ;x-Offset in Cards.

; y-Offset setzen.
          MoveW  clipYPos,r12     ;y-Offset in Pixel.

; Oberste Spalte ausgeben.
          jsr    DrawCurYClip    ;Daten über BitmapClip ausgeben.
          rts                    ;Ende.

;
; Variablen
;
:ScrapName    b "Photo Scrap",NULL

:Flag_ScrapOK b $00      ;$FF = kein Scrap im Speicher.
:Text_NoData  b " * No data * ",NULL

:clipXPos     b $00      ;x-Offset
:clipYPos     w $0000    ;y-offset

```

```

;
; Dialogbox für Anzeige Photoscrap.
;
;
:dbbox      b $01
            b WINPOS_Y
            b WINPOS_Y +WINDOW_Y +3*8 -1
            w WINPOS_X *8
            w WINPOS_X *8 +WINDOW_X *8 -1

            b DB_USR_ROUT
            w InitDBoxMenu

            b DBTXTSTR
            b 8
            b WINDOW_Y +3*8 -8
            w :text1

            b DBTXTSTR
            b 8
            b WINDOW_Y +3*8 -12 -8
            w :text2

            b OK
            b WINDOW_X -6 -1
            b WINDOW_Y +3*8 -16 -8

            b NULL

::text1     b "Laufwerk wählen: Tasten A bis D"
            b NULL

::text2     b "Ausschnitt wählen mit Cursor-Tasten"
            b NULL

```

Zum Abschluss noch ein Screenshot der Programmoberfläche:



Bild L5.1: ScrapViewer

M.6 Demo/Application: "keyData"

Die beiden vorangegangenen Anwendungen "geoScreenCapture" und "ScrapViewer" verwenden auch Tastaturmenüs. Dabei wird der Tastencode in keyData abgefragt.

Um nun nicht jedes mal überlegen zu müssen, welcher Tastencode erforderlich ist um auf eine bestimmte Taste zu reagieren, wurde das Programm "keyData" entwickelt. Das Programm läuft unter GEOS64 und GEOS128, sowohl im 40- als auch im 80-Zeichen-Modus.

Nach dem Start wartet das Programm auf einen Tastendruck. Danach gibt das Programm den Zeichencode, den Dezimalwert und den Hexadezimalwert der Taste auf dem Bildschirm aus. Mit einem Mausklick wird das Programm beendet.

Das Programm kann auch die Tastencodes für Kombinationen mit den Tasten [CTRL] oder [CBM] anzeigen, wobei die Werte sich zwischen der deutschen und der englischen GEOS-Version unterscheiden.

Mit der Hilfe des Programms sind die Tabellen im **Teil D, Anhang L.2 ab Seite 649** dieses Handbuchs entstanden. Das Programm ist daher auch nur als eine Demo-Anwendung zu verstehen, die bei der Arbeit geholfen hat. Für den normalen GEOS-Benutzer dürfte die Anwendung wenig hilfreich sein.

Hier der relative einfache Quelltext:

```
;
; Symboltabellen einbinden.
;
if .p
    t "TopSym"
    t "TopMac"
    t "Sym128.erg"
endif

;
; GEOS-Header definieren.
;
n "keyData"
c "keyData      V1.0"
a "Markus kanet"

f APPLICATION
z $40 ;GEOS64/128 40+80Z.

o APP_RAM

h "Zeichen, Dezimal- und Hex-Wert einer Taste anzeigen."
h "Mausklick=Ende."

i
```



```

;
; Tastaturabfrage starten.
;
;
:MAININIT      php
               sei                      ;Interrupt sperren.

               LoadW r11,$0000
               ldy    #0
;               clc                      ;Nicht nötig wenn r11=$0000
               jsr    StartMouseMode   ;Mausabfrage starten.

               plp                      ;Interrupt freigeben.

::wait         bit    mouseData        ;Warten bis keine
               bpl    :wait            ;Maustaste gedrückt.
               lda    #NULL
               sta    pressFlag

               lda    #0                ;Bildschirm löschen.
               jsr    SetPattern

               ldy    #0
               bit    c128Flag
               bpl    :1
               ldy    #8

::1            ldx    #0
::2            lda    scrData,y
               sta    r2,x
               iny
               inx
               cpx    #6
               bcc    :2

               jsr    Rectangle

               LoadW r11,10
               LoadB r1H,64
               LoadW r0,InfoText
               jsr    PutString

;
; Ende über Mausklick.
; => Rückkehr zum DeskTop.
;
               LoadW otherPressVec,EnterDeskTop

;
; Taste auswerten.
; Kombinationen mit CBM/SHIFT/CTRL
; sind möglich.
               LoadW keyVector,printKey

               rts                      ;Zurück zur GEOS-Mainloop.

```

```

;
; Infotext
;
:InfoText      b PLAINTEXT
                b "Taste drücken für Angaben zu :keyData",CR
                b GOTOX
                w $000a
                b "Zum beenden Maustaste drücken."
                b NULL

;
; Aufruf aus der Mainloop:
; => Taste wurde gedrückt.
;
:printKey      LoadW  r11,10
                LoadB  r1H,20
                jsr    :cleanup          ;Ausgabebereich löschen.

                lda    keyData           ;Taste einlesen.
                cmp    #$20              ;Sichtbare Taste?
                bcc    :1                 ; => Nein, weiter...
                cmp    #$7f              ;Sichtbare Taste?
                bcs    :1                 ; => Nein, weiter...

                LoadW  r11,10            ;Ausgabeposition setzen.
                LoadB  r1H,20
                lda    keyData
                jsr    SmallPutChar       ;Zeichencode ausgeben.
                jsr    :cleanup          ;Ausgabebereich löschen.

::1            LoadW  r11,40             ;Position für
                LoadB  r1H,20            ;Dezimalwert.

                lda    keyData
                sta    r0L
                lda    #$00
                sta    r0H
                lda    #SET_LEFTJUST!SET_SUPRESS
                jsr    PutDecimal         ;Tastencode/Dezimal.

                jsr    :cleanup          ;Ausgabebereich löschen.

                LoadW  r11,70            ;Position für
                LoadB  r1H,20            ;Hexadezimalwert.

                lda    #"$"              ;Hexzahl-Kennung
                jsr    SmallPutChar       ;ausgeben.

                lda    keyData           ;Tastencode nach
                jsr    HEX2ASCII          ;ASCII wandeln.
                pha
                txa
                jsr    SmallPutChar       ;High-Nibble ausgeben.
                pla
                jsr    SmallPutChar       ;Low-Nibble ausgeben.

```



```

::cleanup      lda    #" "          ;Reste von vorheriger
               jsr    SmallPutChar  ;Ausgabe löschen.
               lda    #" "
               jsr    SmallPutChar  ;Proportionalfont!)
               rts

;
; Größe für Bildschirmbereich.
; Beim C128 inkl. Verdoppelung für
; den 80-Zeichen-Bildschirm.
;
:scrData       b 0                ;C64.
               b 199
               w 0
               w 319
               w NULL

               b 0                ;C128.
               b 199
               w 0
               w 319!DOUBLE_W!ADD1_W
               w NULL

;
; HEX-Zahl nach ASCII wandeln.
;
; Übergabe:
; AKKU = Hex-Zahl.
;
; Rückgabe:
; AKKU = Low -Nibble Hex-Zahl.
; XREG = High-Nibble Hex-Zahl.
;
:HEX2ASCII     pha                ;HEX-Wert speichern.
               lsr                ;HIGH-Nibble isolieren.
               lsr
               lsr
               lsr
               jsr    :1          ;HIGH-Nibble nach ASCII.
               tax                ;Ergebnis zwischenspeichern.

               pla                ;HEX-Wert zurücksetzen und
               ;nach ASCII wandeln.

::1            and    #%00001111
               cmp    #10        ;Zahl größer 10?
               bcc    :2        ; => Nein, weiter...
               clc
               adc    #$07        ;Zeichen $A-$F wandeln.

::2            clc
               adc    #"0"
               rts

```

M.7 Demo/Application: "DiskAnalyzerDEMO"

Bei der Überarbeitung dieses Buches sollten auch alle Screenshots neu erstellt werden. Für die Screenshots des "Disk-Analyzer" fehlte zuerst jedoch die passende Anwendung, daher wurde die Oberfläche des Programms nachgebaut und es entstand das Programm "DiskAnalyzerDEMO". Später wurde die GEOS-Application identifiziert, mit deren Hilfe die Screenshots erstellt wurden: Es handelt sich um das Programm "GEOS TOOLS" von W. Knupe, einem Mitautor des Buches.

Hier nun der Quelltext des Nachbaus der Benutzeroberfläche. Der Quelltext ist keine exakte Kopie des Originals, das Programm soll nur den Inhalt des ausgewählten Track/Sektor auf dem Bildschirm darstellen. Allerdings kann der Quelltext auch als einfaches Beispiel für eine GEOS-Application dienen. Fehlende Funktionen kann der findige GEOS-Programmierer selbst ergänzen.

```
;
; Symboldateien einbinden.
;
if .p
    t "TopSym"
    t "TopMac"
endif

;
; GEOS-Header definieren.
;
n "DiskAnalyzerDEMO"
c "ANALYZER V1.0", NULL
a "Markus Kanet", NULL

f APPLICATION
z $80 ;Nur GEOS64.

o APP_RAM

h "Original-Programm GEOSTOOLS von W.Knupe (w)1989"
h "UI-Nachbau von M.Kanet (w)2022"

i


;
; Hauptmenü anzeigen
;
;
:MAININIT    lda    #0                ; Bildschirm löschen.
             jsr    SetPattern

             jsr    i_Rectangle
             b      0,199
             w      0,319
```

```

LoadW  r0,geosmenu      ; GEOS-Menü anzeigen.
jsr     DoMenu

lda     #ST_FLASH
sta     iconSelFlag      ; Icons beim anklicken invertieren.

LoadW  r0,iconmenu      ; ICON-Menü anzeigen.
jsr     DoIcons

jsr     UseSystemFont    ; Systemzeichensatz aktivieren.

LoadW  r0,menutext      ; Menü-Text ausgeben.
jsr     PutString

jsr     Zeige_Adresse    ; Aktuellen Track/Sektor anzeigen.

jsr     Lade_Sektor      ; Sektor-Inhalt einlesen.

jsr     Zeige_Inhalt     ; Sektor-Inhalt anzeigen.

rts     ; Zurück zur GEOS-Mainloop.

;
; Nach BASIC wechseln.
;
:Starte_BASIC LoadW  r0,:Befehl      ; Zeiger auf BASIC-Befehl.

lda     #$00             ; Keine Datei laden.
sta     r5L
sta     r5H

sta     $0800            ; Kein Programm starten.
sta     $0801
sta     $0802
sta     $0803

LoadW  r7,$0803          ; Endadresse setzen.

jmp     ToBasic           ; Nach BASIC wechseln.

; Dummy-Befehlstring für BASIC V2.
::Befehl  b "PRINT"
          b 34,"HELLO WORLD!",34
          b NULL

;
; Sektor einlesen.
;
:Lade_Sektor lda     Adr_TR          ; Aktueller Track/Sektor
sta     r1L              ; nach r1L/r1H.
lda     Adr_SE
sta     r1H

LoadW  r4,diskBlkBuf      ; Sektor nach diskBlkBuf.

```

```

        jsr    GetBlock          ; Sektor einlesen.
        txa
        beq    :ok              ; Diskfehler?
                                   ; => Nein, weiter...

        jmp    Panic            ; Diskfehler!

::ok      rts                    ; Ende...

;
; Track/Sektor anzeigen.
;
;Zeige_Adresse jsr    i_GraphicsString

        b NEWPATTERN            ;Füllmuster setzen.
        b $00

        b MOVEPENTO            ;Zeiger auf xl/yo setzen.
        w $0110
        b $b0

        b RECTANGLETO          ;Rechteck nach xr/yu.
        w $013f
        b $c7

        b NULL                  ;Ende.

        LoadW r0,textcursek     ; Info-Text ausgeben.
        jsr    PutString

        LoadW r11,$012f         ; Cursor setzen.
        LoadB r1H,$b6

        lda    Adr_TR           ; Track nach r0L.
        sta    r0L
        lda    #$00             ; Highbyte immer $00.
        sta    r0H

        ; Max. 12 Pixel breit, rechtsbündig, keine führende 0.
        lda    #12!SET_RIGHTJUST!SET_SUPRESS
        jsr    PutDecimal       ; Track-Adresse ausgeben.

        LoadW r11,$012f         ; Cursor setzen.
        LoadB r1H,$c0

        lda    Adr_SE           ; Sektor nach r0L.
        sta    r0L
        lda    #$00             ; Highbyte immer $00.
        sta    r0H

        ; Max. 12 Pixel breit, rechtsbündig, keine führende 0.
        lda    #12!SET_RIGHTJUST!SET_SUPRESS
        lda    #12!SET_RIGHTJUST!SET_SUPRESS
        jmp    PutDecimal       ; Sektor-Adresse ausgeben.

```

```
;
; Track +1 lesen
;
;
:SetTAdrP1    inc    Adr_TR
              jsr    Zeige_Adresse
              rts

;
; Track -1 lesen
;
;
:SetTAdrM1    dec    Adr_TR
              jsr    Zeige_Adresse
              rts

;
; Sektor +1 lesen
;
;
:SetSAdrP1    inc    Adr_SE
              jsr    Zeige_Adresse
              rts

; Sektor -1 lesen
:SetSAdrM1    dec    Adr_SE
              jsr    Zeige_Adresse
              rts

;
; Track +1 setzen
;
;
:SetTrP1      inc    Adr_TR
              jsr    Zeige_Adresse
              jsr    Lade_Sektor
              jsr    Zeige_Inhalt
              rts

;
; Track -1 setzen
;
;
:SetTrM1      dec    Adr_TR
              jsr    Zeige_Adresse
              jsr    Lade_Sektor
              jsr    Zeige_Inhalt
              rts

;
; Sekor lesen und anzeigen
;
;
:RdPrntSek    jsr    Lade_Sektor
              jsr    Zeige_Inhalt
              rts
```

```

;
; Folgesektor lesen
;
:RdNextSek    lda    diskBlkBuf +0    ;Folgesektor verfügbar?
              beq    :exit            ; => Nein, Ende...
              sta    Adr_TR
              lda    diskBlkBuf +1
              sta    Adr_SE
              jsr    Zeige_Adresse
              jsr    Lade_Sektor
              jsr    Zeige_Inhalt
:exit         rts

;
; HEX-Zahl nach ASCII wandeln.
;
; Übergabe:
; AKKU = Hex-Zahl.
; Rückgabe:
; AKKU = Low-Nibble Hex-Zahl.
; XREG = High-Nibble Hex-Zahl.
:HEX2ASCII    pha                ; HEX-Wert speichern.
              lsr                ; HIGH-Nibble isolieren.
              lsr
              lsr
              jsr    :1            ; HIGH-Nibble nach ASCII wandeln.
              tax                ; Ergebnis zwischenspeichern.

              pla                ; HEX-Wert zurücksetzen und
                                ; nach ASCII wandeln.
:1            and    #%00001111
              clc
              adc    #"0"
              cmp    #"9" +1      ; Zahl größer 10?
              bcc    :2            ; => Ja, weiter...
              clc                ; HEX-Zeichen nach $a-$f wandeln.
              adc    #$27
:2            rts

;
; Sektorinhalt anzeigen
;
:Zeige_Inhalt jsr    i_GraphicsString
              b NEWPATTERN        ;Füllmuster setzen.
              b $00

              b MOVEPENT0        ;Zeiger auf xl/yo setzen.
              w $0000
              b $10

              b RECTANGLETO      ;Rechteck nach xr/yo.
              w $013f
              b $af

```

```

        b NULL                ;Ende.
LoadB  a0L,0                 ; Zeiger auf Byte #1.
LoadW  a1,diskBlkBuf         ; Zeiger auf Datenpuffer.

LoadB  r1H,20                ; Startwert für y-Position Text.

::loop    LoadW  r11,$0000    ; Startwert für x-Position Zeile.
          AddVB  9,r1H        ; y-Position auf nächste Zeile.

          lda    a0L          ; Position einlesen.
          jsr    HEX2ASCII    ; Nach ASCII wandeln.
          pha
          txa
          jsr    SmallPutChar ; High-Nibble ausgeben.
          pla
          jsr    SmallPutChar ; Low-Nibble ausgeben.

          lda    a0L          ; Aktuelles Byte zwischenspeichern.
          pha

          LoadW  r11,13       ; x-Position für HEX-Werte setzen.
          LoadB  a0H,16       ; Anzahl Werte.
          jsr    zeige_hex    ; HEX-Werte anzeigen.

          pla                ; Position wieder auf aktuelles
          sta    a0L          ; Byte zurücksetzen.

          LoadW  r11,190      ; x-Position für ASCII-Werte
          setzten.
          LoadB  a0H,16       ; Anzahl Werte.
          jsr    zeige_ascii  ; ASCII-Werte anzeigen.

          lda    a0L          ; Alle Werte ausgegeben ?
          bne    :loop        ; => Nein, weiter...
          rts

;
; HEX-Werte anzeigen.
;
:zeige_hex    MoveW  r11,a2    ; x-Position zwischenspeichern.

          ldy    a0L
          lda    (a1L),y      ; Aktuelles Zeichen einlesen und
          jsr    HEX2ASCII    ; nach ASCII wandeln.
          pha
          txa
          jsr    SmallPutChar ; High-Nibble ausgeben.
          pla
          jsr    SmallPutChar ; Low-Nibble ausgeben.

          inc    a0L          ; Alle Bytes ausgegeben ?
          beq    :end         ; => Ja, Ende...

          dec    a0H          ; Alle Werte in Zeile ausgegeben ?
          beq    :end         ; => Ja, Ende...

```

```

        lda    a2L                ; x-Position für nächsten Wert
        clc                      ; berechnen.
        adc    #< 11
        sta    r11L
        lda    a2H
        adc    #> 11
        sta    r11H

        jmp    zeige_hex          ; Nächsten Wert ausgeben.
::end
        rts

;
; ASCII-Zeichen anzeigen.
;
; zeige_ascii    MoveW    r11,a2

        ldy    a0L
        lda    (a1L),y           ; Aktuelles Zeichen einlesen.
        cmp    #$20              ; Zeichen < $20 ?
        bcc    :dot              ; => Ja, durch "." ersetzen.
        cmp    #$80 +1          ; Zeichen gültig ?
        bcc    :print            ; => Ja, Zeichen ausgeben.
        lda    #", "            ; Ersatzzeichen.
::dot    jsr    SmallPutChar      ; ASCII-Zeichen ausgeben.
::print

        inc    a0L              ; Alle Bytes ausgegeben ?
        beq    :end             ; => Ja, Ende...
        dec    a0H              ; Alle Werte in Zeile ausgegeben ?
        beq    :end             ; => Ja, Ende...

        lda    a2L                ; x-Position für nächsten Wert
        clc                      ; berechnen.
        adc    #< 8
        sta    r11L
        lda    a2H
        adc    #> 8
        sta    r11H

        jmp    zeige_ascii       ; Nächsten Wert ausgeben.
::end
        rts

;
; Programmdaten.
;
; Adr_TR        b $07
; Adr_SE        b $03

;
; Texte für Hauptmenü.
;
; menutext      b PLAINTEXT
;               b GOTOXY
;               w $0011
;               b $bb
;               b "Track"

```



```

        b GOTOXY
        w $0051
        b $bb
        b "Sektor"

:textcursek    b PLAINTEXT
               b GOTOXY
               w $0112
               b $b6
               b "Track:"

               b GOTOXY
               w $0112
               b $c0
               b "Sektor:"

               b NULL

;
; Hauptmenü
;
:geosmenu      b 0 ,14
               w 0 ,319
               b 2 ! HORIZONTAL ! UN_CONSTRAINED

               w :01
               b MENU_ACTION
               w EnterDeskTop

               w :02
               b MENU_ACTION
               w Starte_BASIC

::01           b "GEOS", NULL
::02           b "BASIC", NULL

;
; Icon-Menü.
;
:iconmenu      b 10
               w $0000
               b $00

               w icon_plus
               b $00,$b0,icon_plus_x,icon_plus_y
               w SetTAdRP1

               w icon_minus
               b $05,$b0,icon_minus_x,icon_minus_y
               w SetTAdRM1

               w icon_plus
               b $08,$b0,icon_plus_x,icon_plus_y
               w SetSAdRP1

```

```

        w icon_minus
        b $0e,$b0,icon_minus_x,icon_minus_y
        w SetSAdrM1

        w icon_r
        b $11,$b0,icon_r_x,icon_r_y
        w RdPrntSek

        w icon_n
        b $14,$b0,icon_n_x,icon_n_y
        w RdNextSek

        w icon_m
        b $17,$b0,icon_m_x,icon_m_y
        w $0000

        w icon_s
        b $1a,$b0,icon_s_x,icon_s_y
        w $0000

        w icon_t_minus
        b $1d,$b0,icon_t_minus_x,icon_t_minus_y
        w SetTrM1

        w icon_t_plus
        b $20,$b0,icon_t_plus_x,icon_t_plus_y
        w SetTrP1
;

; Dummy-Icon
if 0
; Packer-Code $80 + 32 Byte ungepackte Grafikdaten
:icon_dummy    b $80 +32
                b %00000000,%00000000
                b %01111111,%11111110
                b %01000000,%00000011
                b %01000000,%00000011
                b %01000000,%00000011
                b %01000000,%00000011
                b %01000000,%00000011
                b %01000000,%00000011
                b %01000000,%00000011
                b %01000000,%00000011
                b %01000000,%00000011
                b %01000000,%00000011
                b %01000000,%00000011
                b %01000000,%00000011
                b %01111111,%11111111
                b %00111111,%11111111

:icon_dummy_x = 2
:icon_dummy_y = 16
endif

```

```
;  
;  
; Menü-Icons.  
;  
;  
; Packer-Code $80 + 32 Byte ungepackte Grafikdaten  
:icon_plus      b $80 +32  
                 b %00000000,%00000000  
                 b %01111111,%11111110  
                 b %01000000,%00000001  
                 b %01000000,%00000011  
                 b %01000001,%10000011  
                 b %01000001,%10000011  
                 b %01000001,%10000011  
                 b %01001111,%11110011  
                 b %01001111,%11110011  
                 b %01000001,%10000011  
                 b %01000001,%10000011  
                 b %01000001,%10000011  
                 b %01000000,%00000011  
                 b %01000000,%00000011  
                 b %01111111,%11111111  
                 b %00111111,%11111111  
  
:icon_plus_x = 2  
:icon_plus_y = 16  
  
;  
;  
; Packer-Code $80 + 32 Byte ungepackte Grafikdaten  
:icon_minus     b $80 +32  
                 b %01111111,%11111110  
                 b %01000000,%00000011  
                 b %01000000,%00000011  
                 b %01000000,%00000011  
                 b %01000000,%00000011  
                 b %01000000,%00000011  
                 b %01001111,%11110011  
                 b %01001111,%11110011  
                 b %01000000,%00000011  
                 b %01000000,%00000011  
                 b %01000000,%00000011  
                 b %01000000,%00000011  
                 b %01000000,%00000011  
                 b %01111111,%11111111  
                 b %00111111,%11111111  
  
:icon_minus_x = 2  
:icon_minus_y = 16
```

```
;
; Packer-Code $80 + 32 Byte ungepackte Grafikdaten
:icon_r      b $80 +32
              b %00000000,%00000000
              b %01111111,%11111110
              b %01000000,%00000011
              b %01011111,%11110011
              b %01011111,%11110011
              b %01011000,%00011011
              b %01011000,%00011011
              b %01011000,%00011011
              b %01011111,%11111011
              b %01011111,%11110011
              b %01011000,%01100011
              b %01011000,%00110011
              b %01011000,%00011011
              b %01000000,%00000011
              b %01111111,%11111111
              b %00111111,%11111111

:icon_r_x = 2
:icon_r_y = 16

;
; Packer-Code $80 + 32 Byte ungepackte Grafikdaten
:icon_n      b $80 +32
              b %00000000,%00000000
              b %01111111,%11111110
              b %01000000,%00000011
              b %01011110,%00011011
              b %01011110,%00011011
              b %01011011,%00011011
              b %01011011,%00011011
              b %01011001,%10011011
              b %01011001,%10011011
              b %01011000,%11011011
              b %01011000,%11011011
              b %01011000,%01111011
              b %01011000,%01111011
              b %01000000,%00000011
              b %01111111,%11111111
              b %00111111,%11111111

:icon_n_x = 2
:icon_n_y = 16
```

```

;
; Packer-Code $80 + 32 Byte ungepackte Grafikdaten
:icon_m      b $80 +32
              b %00000000,%00000000
              b %01111111,%11111110
              b %01000000,%00000011
              b %01011110,%01111011
              b %01011110,%01111011
              b %01011011,%11011011
              b %01011011,%11011011
              b %01011001,%10011011
              b %01011001,%10011011
              b %01011000,%00011011
              b %01011000,%00011011
              b %01011000,%00011011
              b %01011000,%00011011
              b %01000000,%00000011
              b %01111111,%11111111
              b %00111111,%11111111

:icon_m_x = 2
:icon_m_y = 16

;
; Packer-Code $80 + 32 Byte ungepackte Grafikdaten
:icon_s      b $80 +32
              b %00000000,%00000000
              b %01111111,%11111110
              b %01000000,%00000011
              b %01001111,%11111011
              b %01011111,%11111011
              b %01011000,%00000011
              b %01011000,%00000011
              b %01011111,%11111011
              b %01001111,%11111011
              b %01000000,%00011011
              b %01000000,%00011011
              b %01011111,%11111011
              b %01011111,%11111011
              b %01000000,%00000011
              b %01111111,%11111111
              b %00111111,%11111111

:icon_s_x = 2
:icon_s_y = 16

```

```
;  
; Packer-Code $80 + 32 Byte ungepackte Grafikdaten  
:icon_t_minus b $80 +32  
               b %00000000,%00000000  
               b %01111111,%11111110  
               b %01000000,%00000011  
               b %01011111,%11111011  
               b %01011111,%11111011  
               b %01000001,%10000011  
               b %01000001,%10000011  
               b %01000001,%10000011  
               b %01000001,%10000011  
               b %01000001,%10000011  
               b %01000001,%10111011  
               b %01000001,%10000011  
               b %01000000,%00000011  
               b %01111111,%11111111  
               b %00111111,%11111111  
  
:icon_t_minus_x = 2  
:icon_t_minus_y = 16  
  
;  
; Packer-Code $80 + 32 Byte ungepackte Grafikdaten  
:icon_t_plus  b $80 +32  
               b %00000000,%00000000  
               b %01111111,%11111110  
               b %01000000,%00000011  
               b %01011111,%11111011  
               b %01011111,%11111011  
               b %01000001,%10000011  
               b %01000001,%10000011  
               b %01000001,%10000011  
               b %01000001,%10010011  
               b %01000001,%10111011  
               b %01000001,%10010011  
               b %01000001,%10000011  
               b %01000000,%00000011  
               b %01111111,%11111111  
               b %00111111,%11111111  
  
:icon_t_plus_x = 2  
:icon_t_plus_y = 16
```

M.8 Systemroutinen: "EnableIO / DisableIO"

Um unter GEOS64 auf den I/O-Bereich zuzugreifen können die Routinen *InitForIO* und *DoneWithIO* verwendet werden. Bei Verwendung einer CMD-SuperCPU oder eines TurboChameleon64 wird dabei allerdings auch der Prozessortakt des C64 auf 1MHz herunter geregelt.

Sollen nur I/O-Register verändert werden und nicht gleichzeitig auf den seriellen Bus zugegriffen werden, dann reicht es aus nur das Prozessorregister zu ändern.

```
; I/O-Bereich einblenden, Interrupt blockieren.
:EnableIO      php
               sei

               pla
               sta     IRQ_RegBuf      ;Interrupt-Status speichern.

               lda     CPU_DATA
               sta     CPU_RegBuf      ;Prozessorregister speichern.
               lda     #IO_IN
               sta     CPU_DATA

               rts

; Speicherkonfiguration und Interrupt-Status zurücksetzen
:DisableIO     lda     CPU_RegBuf      ;Prozessorregister zurücksetzen.
               sta     CPU_DATA

               lda     IRQ_RegBuf      ;Interrupt-Status zurücksetzen.
               pha

               plp
               rts

:IRQ_RegBuf    b $00
:CPU_RegBuf    b $00
```

Wie auch bei *InitForIO* und *DoneWithIO* dürfen diese Routinen nicht geschachtelt werden, da nur der zuletzt aktive Zustand des Prozessors gesichert und wieder zurückgesetzt wird. Ruft man *EnableIO* (oder *InitForIO*) mehrfach hintereinander auf, dann kann dies zu einem Systemabsturz führen wenn *DisableIO* (bzw. *DoneWithIO*) aufgerufen wird.

GEOS-Programmierung mit dem MegaAssembler

»Teil D«

Erweiterungen für GEOS und MegaAssembler

Bereits in den 1990er-Jahren wurde GEOS und der MegaAssembler überarbeitet. Teil D des Buches enthält eine Vielzahl an Fehlerkorrekturen und zusätzliche Informationen vom Entwickler der Erweiterungen, um diese sinnvoll für eigene Projekte nutzen zu können.

Kurzübersicht über alle GEOS-Routinen

Im Anhang befindet sich eine neue Kurzübersicht über alle GEOS-Routinen inkl. Angabe welche Parameter benötigt werden und welche Register verändert werden.