GEOS - MegaPatch64

Technische Dokumentation für Programmierer

Stand: 23.11.2019 für MegaPatch V3.3r6

Inhaltsübersicht:

(1) EINLEITUNG

Das Programm "GEOS-MegaPatch" entstand aus der Not des Programmierers:

Viele Funktionen innerhalb des Kernals waren fehlerhaft oder bestimmte, häufig benötigte Funktionen waren umständlich zu Programmieren.
Außerdem gibt es schon seit langer Zeit viele unterschiedliche Patches für GEOS, so daß kaum ein Anwender das gleiche "Kernal" wie ein anderer GEOS-Anwender besitzt. Inkompatibilitäten waren an der Tagesordnung und der Programmierer war gezwungen seine Programme auf allen möglichen Systemen zu testen. Die Fehlersuche konnte länger dauern als die Programmentwicklung.

Deshalb entstand die Idee des GEOS-MegaPatch: Alle bisherigen Ideen für Verbesserungen am GEOS-System wurden aufgegriffen, verbessert und in ein GEOS-Programm umgesetzt. Das Ergebnis ist das wohl beste GEOS-Patch das es für den C64 je gegeben hat.

Mit MegaPatch entfallen nun die verschiedenen Startprogramme für jede Speichererweiterung. Auch die bisherigen Schwachstellen, wie etwa das bei einer RAMCard keine RAMLink-Partitionen verwendet werden dürfen, wurden beseitigt. Auch dieser Punkt beruhte auf diversen inkompatiblen Patches für "Configure".

Das MegaPatch-Programm erlaubt nun allen GEOS-Programmierern wieder auf eine feste Konfiguration hin zu programmieren. Es müssen also keinerlei Anpassungen mehr für verschiedene Systeme entwickelt werden (wie bisher für GateWay und GEOS).

(2) GEÄNDERTE GEOS-REGISTER

Adresse ":driveType" ($848E)

Dieses Register wurde neu definiert: Es enthält jetzt den Emulationsmodus des aktuellen Laufwerks. Damit wird sichergestellt, das bestehende Programme keine Probleme mit den neuen Laufwerkstreibern und Laufwerkstypen haben.

Tabelle 1: Emulationsformate

Emulation	Typ	Kennbyte
1541	1541, FD41, HD41, SD2IEC	$01
1541-Cache	1541	$41
1571	1571, FD71, HD71, SD2IEC	$02
1581	1581, FD81, HD81, SD2IEC	$03
Native	FD-Native, HD-Native, SD2IEC-Native	$04
PCDOS	1581-DOS, FD-DOS	$05
RAM41	RAM1541, RAMLink1541	$81
RAM71	RAM1571, RAMLink1571	$82
RAM81	RAM1581, RAMLink1581	$83
RAMNative	RAMNative, RAMLinkNative	$84

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

Adresse ":ramBase" ($88C7)

Um 100% kompatibel zum Original GEOS-V2 zu bleiben wurde dieses Register nicht verändert. Auch weiterhin erfährt man hier die Startadresse des aktuellen RAM-Laufwerks in der Speichererweiterung.

Eine Sonderstellung nimmt hier die RAMLink ein: In GEOS-V2 findet man in :ramBase+X das HIGH-Byte der Startpartition und in :driveData+3 das zugehörige LOW-Byte. Die aktive Partition verwalten die RAMLink-Treiber nun intern und sind daher nicht mehr auf diese Adressen angewiesen.
Um aber auch weiterhin den Partitionswechsel über :ramBase zu erlauben prüft der RAMLink-Treiber das HIGH-Byte auf Veränderungen. Findet der Treiber hier einen neuen Wert (z.B. nachdem "CMD_Move" verwendet wurde), so wird die zugehörige Partition auf der RAMLink gesucht und geöffnet. Das Register :driveDat+3 wird nicht mehr benötigt.
Damit ist es möglich auch als Laufwerk D: ein 1571-Laufwerk zu installieren. Bisher war das nicht möglich, da die 1571-Treiber in :driveData Informationen über die aktuelle Diskette ablegen ($00=1541, $80=1571).

(3) BESCHREIBUNG DER NEUEN GEOS-REGISTER

Adresse ":DskDrvBaseL" ($9F7E, 4 Bytes)
Adresse ":DskDrvBaseH" ($9F82, 4 Bytes)

Diese Adressen dienen der internen Verwaltung der Laufwerkstreiber im erweiterten Speicher. In :DiskDrvBaseL finden sich die Low-Bytes der Adressen für Laufwerk A: bis D:, in :DiskDrvBaseH die dazugehörigen High-Bytes.

Adresse ":doubleSideFlg" ($9F86, 4 Bytes)

Diese Adressen geben Auskunft ob die aktuelle Diskette in einem 1571-Laufwerk einseitig ($00) oder doppelseitig ($80) ist. Diese Werte wurden früher in :driveData abgelegt. Dieses Register wurde von CMD jedoch unglücklicherweise zur RAMLink-Verwaltung herangezogen so daß Doppelbelegungen vorkommen konnten.

Adresse ":drivePartData" ($9F8A, 4 Bytes)

Diese Adressen enthalten nach dem öffnen einer Diskette die aktive Partitions-Nr. auf CMD-Laufwerken.

Adresse ":RealDrvType" ($9F8E, 4 Bytes)

Diese Adressen geben detaillierte Auskunft über das angeschlossene Laufwerk:

Tabelle 2: Laufwerkstypen

Laufwerkstyp	Kennung	Anmerkungen
Commodore 1541 / SD2IEC-1541	$01	
Commodore 1571 / SD2IEC-1571	$02	
Commodore 1581 / SD2IEC-1581	$03	
SD2IEC-Native / IECBUS-Native	$04	GEOS-MegaPatch ab 2018.
Commodore 1581 - DOS-Modus	$05	
CMD FD2000/4000 1541-Modus	$11	
CMD FD2000/4000 1571-Modus	$12	
CMD FD2000/4000 1581-Modus	$13	
CMD FD2000/4000 Native-Modus	$14	
CMD FD2000/4000 DOS-Modus	$15	
CMD HD 1541-Modus	$21	
CMD HD 1571-Modus	$22	
CMD HD 1581-Modus	$23	
CMD HD Native-Modus	$24	
CMD RAMLink 1541-Modus	$31	
CMD RAMLink 1571-Modus	$32	
CMD RAMLink 1581-Modus	$33	
CMD RAMLink Native-Modus	$34	
Commodore 1541 mit RAM-Cache	$41	
RAM1541	$81	
RAM1571	$82	
RAM1581	$83	
RAMNative	$84	
C=REU-Native	$a4	GEOS-MegaPatch ab 2018.
GeoRAM-Native	$b4	GEOS-MegaPatch ab 2018.
SuperRAM-Native	$c4	

Damit ist es möglich, das aktuelle Laufwerk korrekt zu bestimmen. Weitere Laufwerkstypen können noch definiert werden. Man sollte jedoch darauf achten, das man keine Laufwerkstypen doppelt belegt und das einige Programme über Prüfung bestimmter Bit-Werte Laufwerke erkennen.

HINWEIS: SD2IEC-Laufwerke benötigen im 1541/71/81-Modus die "file-based M-R emulation", also ein Laufwerks-ROM. Dazu reicht es aus ein ROM-File auf das SD2IEC zu kopieren (ROOT/Haupt-Verzeichnis) und via "XR"-Befehl zu aktivieren. Es empfiehlt sich folgende Dateinamen zu verwenden:

SD2IEC/D64-Image	DOS1541.BIN
SD2IEC/D71-Image	DOS1571.BIN
SD2IEC/D81-Image	DOS1581.BIN

Adresse ":RealDrvMode" ($9F92, 4 Bytes)

Diese Adressen geben Auskunft über erweiterte Fähigkeiten für das angeschlossene Laufwerk. Folgende Möglichkeiten sind unter MegaPatch vorgesehen:

Tabelle 3: Laufwerkseigenschaften

Bit	Variable	Laufwerksfähigkeit
%1xxxxxxx	:SET_MODE_PARTITION	Partitionen
%x1xxxxxx	:SET_MODE_SUBDIR	Unterverzeichnisse
%xx1xxxxx	:SET_MODE_FASTDISK	Schnelles Laufwerk (RAM / CMDHD+PP-Kabel)
%xxx1xxxx	:SET_MODE_SRAM	SuperRAM-Laufwerk
%xxxx1xxx	:SET_MODE_CRAM	C=REU-Laufwerk
%xxxxx1xx	:SET_MODE_GRAM	GeoRAM-Laufwerk
%xxxxxx1x	:SET_MODE_SD2IEC	SD2IEC-Laufwerk

Damit kann die Abfrage des aktuellen Laufwerkstyps entfallen. Man prüft einfach das entsprechende Bit in ":RealDrvMode" und ruft entsprechende Unterprogramme auf.
Die Eigenschaften "SET_MODE_xRAM" definieren RAM-Laufwerke die außerhalb des GEOS-DACC im erweiterten Speicher angelegt wurden. Diese Laufwerke nutzen den Speicheroberhalb der 4Mb-Grenze die MegaPatch selbst als Speichererweiterung verwendet. Die Laufwerke für C=REU und GeoRAM wurden in der Version von 2018 in MegaPatch ergänzt.
"SET_MODE_SD2IEC" wurde 2019 ab Version V3.3r4 ergänzt und kennzeichnet ein SD2IEC-Laufwerk in Verbindung mit dem 1541/71/81 oder SD2IEC-Laufwerkstreiber.

Adresse ":RamBankInUse" ($9F96, 16 Bytes)

Diese Tabelle zeigt an welche der verfügbaren Speicherbänke frei oder belegt sind. Jeder Speicherbank sind dabei zwei Bit zugeordnet. Innerhalb eines Bytes werden die Bänke von links nach rechts (beginnend mit dem höchsten Bit %7) durchnummeriert. Beispiel:

Tabelle 4: Schlüssel für Bankbelegungstabelle

Bit-Paar	64K-Speicherbank
%11xxxxxx	Bank #0
%xx11xxxx	Bank #1

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

Tabelle 5: Belegungsmodi

Bit-Paar	64K-Speicherbank
%00	Speicherbank frei
%01	Durch Anwendungen belegt
%10	Durch Laufwerkstreiber belegt
%11	Durch GEOS, Spooler, TaskManager oder aktive Anwendungen belegter Speicher. Dieser Speicher kann nur durch einen Neustart oder durch die einzelnen Anwendungen selbst wieder freigegeben werden.

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

```
* CODE *
:FndFreeBnk 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 ($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.
```

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 oder zwischenzeitlich die MainLoop abläuft und der TaskManager gestartet werden kann, der muß hier die Speicherbank als belegt kennzeichnen (am besten mit $11). Man läuft sonst Gefahr das eine andere Anwendung die gleiche Speicherbank als "Frei" erkennt und für eigene Zwecke verwendet.

Max. stehen hier 16 Bytes für max. 64 Speicherbänke = 4MByte zur Verfügung. über :ramExpSize kann man feststellen ob die Speicherbank überhaupt verfügbar ist.
Bei der SuperCPU existiert noch die Möglichkeit einen Teil des RAMCard-Speichers zu verwenden. Welche Bereiche frei sind, kann man über die entsprechenden RAMCard-Register der SuperCPU erfahren.

Adresse ":RamBankFirst" ($9FA6, 1 Word)

Diese Adresse definiert die Startadresse des GEOS-DACC-Speichers in der Speichererweiterung.
Bei einer GeoRAM oder C=REU ist hier immer der Wert $0000 zu finden.
Bei der RAMLink findet man hier die Startadresse der aktiven DACC-Partition.
Bei einer RAMCard findet man hier die erste freie Speicherbank.

Adresse ":GEOS_RAM_TYP" ($9FA8, 1 Byte)

Informationen zur aktuellen Speichererweiterung. :GEOS_RAM_TYP kann folgende Werte annehmen:

RAM_SCPU	$10
RAM_BBG	$20
RAM_REU	$40
RAM_RL	$80

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

```
* CODE *     lda   GEOS_RAM TYP
             cmp   #RAM_SCPU
             bne   ERROR
```

In diesem Beispiel wird das Programm beendet, wenn keine RAMCard unter MegaPatch eingesetzt wird.

Adresse ":MP3_64K_SYSTEM" ($9FA9, 1 Byte)

Hier findet man die 64K-Bank welche das erweiterte MegaPatch-Kernal beinhaltet. Dieses ist in der Regel die letzte verfügbare GEOS-Speicherbank innerhalb des GEOS-DACC.

Adresse ":MP3_64K_DATA" ($9FAA, 1 Byte)

In dieser Speicherbank lagert MegaPatch verschiedene Daten aus wie z.B. das SwapFile oder die Hintergrundgrafik.

Adresse ":MP3_64K_DISK" ($9FAB, 1 Byte)

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

Adresse ":Flag_Optimize" ($9FAC, 1 Byte)

Findet man hier den Wert $00, so wird die SuperCPU für MegaPatch optimiert. Für einige Programme muß die Optimierung jedoch deaktiviert werden (geoBASIC und GeoProgrammer/Debugger). Dies kann man über den GEOS.Editor erledigen. Der Wert $03 schaltet die Optimierung für GEOS aus.

Adresse ":millenium" ($9FAD, 1 Byte)

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

Adresse ":Flag_LoadPrnt" ($9FAE, 1 Byte)

Dieses Flag kann über den GEOS.Editor geändert werden: Ist der Wert $00, dann wird der Druckertreiber immer von Diskette gestartet. Hat :Flag_LoadPrnt den Wert $80, so wird der Druckertreiber aus dem RAM geladen.

Adresse ":PrntFileNameRAM" ($9FAF, 17 Bytes)

Hier findet man den Namen des im RAM gespeicherten Druckertreibers. Dieser muß nicht mit :PrntFileName ($8465) übereinstimmen, da andere Anwendungsprogramme den Druckertreiber wechseln, indem hier der Name des neuen Treibers eingetragen wird.
Die neue GetFile-Routine von MegaPatch vergleicht dann :PrntFileName mit :PrntFileNameRAM und prüft ob ein neuer Druckertreiber aktiviert werden soll. Danach wird der neue Druckertreiber geladen und 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, den Rest erledigt dann MegaPatch.

Adresse ":Flag_Spooler" ($9FC0, 1 Byte)

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

Adresse ":Flag_SpoolMinB" ($9FC1, 1 Byte)

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

Adresse ":Flag_SpoolMaxB" ($9FC2, 1 Byte)

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.

Adresse ":Flag_SpoolADDR" ($9FC3, 3 Bytes)

Zeiger auf die aktuelle Position im Spooler-RAM. An diese Position wird das nächste Byte geschrieben. Format: LOW-Byte, HIGH-Byte, BANK-Adresse.

Adresse ":Flag_SpoolCount" ($9FC6, 1 Byte)

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 Menü kann dann nur über den TaskManager gestartet werden (<C=> + <CTRL>, Register "Drucker/Druckerspooler").

Adresse ":Flag_Sp1CurDok" ($9FC7, 1 Byte)

Angabe der Dokument-Nummer in der Spooler-Warteschlange.

Adresse ":Flag_Sp1MaxDok" ($9FC8, 1 Byte)

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

Adresse ":Flag_TaskAktiv" ($9FC9, 1 Byte)

Der Wert $00 signalisiert "TaskManager aktiviert". Wann immer es notwendig ist, das der TaskManager nicht gestartet werden darf (z.B. bei komplexen Diskettenoperationen) kann man hier den Wert $FF eintragen. MegaPatch übergeht dann die Tastaturabfrage zur Aktivierung des TaskManagers. Man sollte allerdings den Originalinhalt des Register zwischenspeichern und später wieder zurückschreiben.

Adresse ":Flag_TaskBank" ($9FCA, 1 Byte)

Systemspeicherbank für TaskManager. In dieser Bank findet man die aktuelle Version des TaskManagers. Diese wird beim Aufruf in den Speicher geholt und gestartet.

Adresse ":Flag_ExtRAMinUse" ($9FCB, 1 Byte)

Verschiedene Speicherbereiche in der REU sind für die Dialogbox und Hilfsmittel (DeskAccessoires) 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 Bereich des SwapFiles der ersten Anwendung zerstört. Bei folgenden Werten (auch Kombinationen) kann der TaskManager nicht in eine andere Anwendung wechseln:

$40	SwapFile aktiv
$80	Dialogbox geoeffnet

Adresse ":Flag_ScrSvCnt" ($9FCC, 1 Byte)

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

Adresse ":Flag_ScrSaver" ($9FCD, 1 Byte)

Initialisierungsbyte für den aktuellen Bildschirmschoner. Die Bedeutung der Werte im einzelnen:

$00	Bildschirmschoner-Effekt starten
$20	Aktivierungszeit herunterzählen
$40	Aktivierungszeit zurücksetzen
$80	Bildschirmschoner deaktivieren

Adresse ":Flag_CrsrRepeat" ($9FCE, 1 Byte)

Dieses Register definiert die Blinkfrequenz des Cursors. Ein Wert von $03 ist optimal, $0F entspricht der Frequenz unter GEOS-V2.

Adresse ":BackScrPattern" ($9FCF, 1 Byte)

Unter MegaPatch haben Anwendungsprogramme die Möglichkeit den Hintergrundfrei zu gestalten. Der Hintergrund kann entweder ein frei wählbares Hintergrundbild sein oder eines der GEOS-Füllmuster.
Wird im GEOS.Editor die Option "Anzeige/Hintergrund/Hintergrundbild verwenden" deaktiviert, so verwendet MegaPatch das hier angegebene GEOS-Füllmuster zum löschen des Bildschirms.

Adresse ":Flag_SetColor" ($9FD0, 1 Byte)

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

$00	Farbe nicht setzen
$40	Farbe nur bei Standard-Dialogbox
$80	Farbe immer setzen

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

Adresse ":Flag_ColorDBox" ($9FD1, 1 Byte)

Dieses Byte wird von der Dialogbox-Routine berechnet. Findet man hier den Wert $80, so werden in der aktuellen Dialogbox keine Farben gesetzt. Diese Funktion bleibt ohne Wirkung wenn es sich um eine Dateiauswahlbox handelt. Diese wird generell in Farbe gezeichnet.

Adresse ":Flag_IconMinX" ($9FD2, 1 Byte)

MegaPatch stellt für alle Icons nun auch Farbe zur Verfügung. Allerdings kann es passieren das kleinere Icons nicht im Format von 8x8 Pixel vorliegen. Deshalb setzt MegaPatch erst dann Farbe für Icons wenn diese eine Mindestgröße besitzen. Standardmäßig müssen Icons mindestens 5 CARDs (40 Pixel) breit sein, damit MegaPatch Farbe aktiviert. Dieser Wert entspricht der Breite der OK, ABBRUCH und ÖFFNEN-Icons. Bei der Dateiauswahlbox wird Farbe für alle Icons gesetzt.

Adresse ":Flag_IconMinY" ($9FD3, 1 Byte)

Das gleiche gilt für die Höhe eines Icons. Standardmäßig findet man hier den Wert $10 = 16 Pixel. Das entspricht ebenfalls den Angaben der Systemicons.

Adresse ":Flag_IconDown" ($9FD4, 1 Byte)

Icons können in X-Richtung nur im Bereich einzelner CARDs positioniert werden, in Y-Richtung gilt diese Einschränkung nicht.
Da de C64 Farbe aber immer nur innerhalb von CARDs setzen kann, muß MegaPatch die Icons verschieben. Je nach Differenz zum nächsten CARD ermittelt MegaPatch den günstigtsen 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.

Adresse ":Flag_DBoxType" ($9FD5, 1 Byte)

Hier findet man nach Aufruf der Dialogbox eine Kopie des Definitionsbytes der Dialogboxtabelle. Wird nur intern vom Kernal benötigt.

Adresse ":Flag_GetFiles" ($9FD6, 1 Byte)

Wenn die Routine :DoDlgBox ($C256) aufgerufen wird, dann untersucht MegaPatch zuerst die gesamte Tabelle nach den Steuercodes DBGETFILES und DBUSRFILES. Wird einer dieser beiden Codes verwendet, dann wird später die externe Dateiauswahlbox gestartet. MegaPatch setzt dann hier ein Flag und unterbindet somit alle Bildschirmausgaben. Lediglich Icons werden noch bearbeitet, da diese für die Auswahlbox benötigt werden.

Adresse ":DB_GFileType" ($9FD7, 1 Byte)

Bei einer Dateiauswahlbox muß man in :r7L den Dateityp ablegen, dieser wird vom GEOS-Kernal nach :DB_GFileType kopiert.

Adresse ":DB_GFileClass" ($9FD8, 1 Word)

Hierher kopiert das Kernal den Zeiger auf die GEOS-Klasse für die Dateiauswahlbox.

Adresse ":DB_GetFileEntry" ($9FDA, 1 Byte)

Dieses Byte zeigt auf den gewählten Eintrag in der Dialogboxtabelle. Für eine Dialogbox über DBGETFILES hat dieser Wert keinerlei Bedeutung, da die Dateinamen nicht im Speicher abgelegt werden. Nur 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 der GEOS.Editor bei der Laufwerksauswahl zunutze: Hier intressiert sich der Editor nicht für den Namen des gewählten Laufwerks, sondern nur für den Wert in DB_GetFileEntry. Dieser zeigt dann auf eine Tabelle mit den verfügbaren Laufwerkstreibern.

Adresse ":DB_StdBoxSize" ($9FDB, 6 Bytes)

Hier findet man die Werte für die Größe einer Standard-Dialogbox (Dialogboxtabelle mit einem KopfByte Typ $8x):

Adresse	Format	Beschreibung
$9FDB	Byte	Obere Grenze
$9FDC	Byte	Untere Grenze
$9FDD	Word	Linke Grenze
$9FDE	Word	Rechte Grenze

Adresse ":Flag_SetMLine" ($9FE1, 1 Byte)

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 dargestellt. Der Wert $80 zeichnet keinerlei Trennlinien.

Adresse ":Flag_MenuStatus" ($9FE2, 1 Byte)

Dieses Register definiert das Aussehen von PullDown-Menüs.
Bit %7 ist für das invertieren des aktuellen Menüeintrages verantwortlich. Ist es gesetzt (Bit %7 = #1), dann wird der aktuelle Eintrag invertiert.
Bit %6 ist für das verlassen der Dialogbox nach unten zuständig. Hat Bit %6 den Wert #1, so kann man Menüs nicht nach unten verlassen. Ist Bit %6 gelöscht, so kann man Menüs auch nach unten verlassen. Dies funktioniert allerdings nur wenn der Programmierer dies generell zuläßt. Ist das verlassen von Menüs nur nach oben möglich, dann bleibt dieses Bit ohne Funktion.

Adresse ":DM_LastEntry" ($9FE3, 6 Bytes)

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

Adresse ":DM_LastNumEntry" ($9FE9, 1 Byte)

Zeigt auf den invertierten Eintrag in der Menütabelle.

Farben für das MegaPatch-System ($9FEA, 22 Bytes)

Fast alle Farben des MegaPatch-Systems werden aus der folgenden Tabelle ausgelesen. Wer spezielle Farben setzen möchte, der kann hier die Werte verändern:

Tabelle 6: Die Farben des MegaPatch-Systems

Adr.	Bezeichnung	Bytes	Funktion
$9FEA	C_Balken	1	Rollbalken
$9FEB	C_Register	1	Aktiver Registerkarten-Reiter
$9FEC	C_RegisterOff	1	Inaktiven Registerkarten-Reiter
$9FED	C_RegisterBack	1	Hintergrund für Registerkarten
$9FEE	C_Mouse	1	Mauszeiger
$9FEF	C_DBoxTitel	1	Dialogbox-Titelzeile
$9FF0	C_DBoxBack	1	Dialogbox-Hintergrund
$9FF1	C_DBoxDIcon	1	Dialogbox-Icons
$9FF2	C_FBoxTitel	1	Dateiauswahlbox-Titelzeile
$9FF3	C_FBoxBack	1	Dateiauswahlbox-Hintergrund
$9FF4	C_FBoxDIcon	1	Dateiauswahlbox-Icons
$9FF5	C_FBoxFiles	1	Dateiauswahlbox-Dateifenster
$9FF6	C_WinTitel	1	Fenster-Titelzeile
$9FF7	C_WinBack	1	Fenster-Hintergrund
$9FF8	C_WinShadow	1	Fenster-Schatten
$9FF9	C_WinIcon	1	Fenster-Icons
$9FFA	C_PullDMenu	1	PullDown-Menüs GEOS-MegaPatch unterstützt von sich aus keine Farbe in PullDown-Menüs. Wer Farbe setzten möchte muß dies über eine eigene Routine erledigen.
$9FFB	C_InputField	1	Eingabefelder
$9FFC	C_InputFieldOff	1	Inaktives Optionsfeld
$9FFD	C_GEOS_BACK	1	GEOS-Hintergrund
$9FFE	C_GEOS_FRAME	1	GEOS-Rahmen
$9FFF	C_GEOS_MOUSE	1	GEOS-Mauszeiger (Kopie von C_Mouse)

HINWEIS: Das Programm "TopDesk" in der Version von 2003 oder 2018 ändert einige der hier aufgeführten Werte um seine eigenen Dialogboxen in der richtigen Farbe darzustellen.

Grundsätzlich sollte auf diese Adressen nur "lesend" zugegriffen werden oder es sollte dem Anwender gegenüber klargestellt werden, das die Änderungen systemweit gelten.

(4) GEÄNDERTE KERNAL-FUNKTIONEN

Adresse ":FindFTypes" ($C23B)

Diese Routine erstellt eine Tabelle mit Dateinamen gleichen Dateityps. Jeder Dateiname hat eine Länge von 16 Zeichen plus einem Null-Byte. In :r7L wird der Dateityp übergeben.
Gegenüber früheren GEOS-Versionen bedeutet hier der Dateityp $FF das alle Dateien des aktuellen Verzeichnisses eingelesen werden sollen.
Die einzige Begrenzung besteht in der Anzahl der Dateien: Diese ist auf max. 255 Dateien begrenzt.

Adresse ":DoDlgBox" ($C256)

Die Definition einer Dialogbox-Tabelle wurde um zusätzliche Funktionen erweitert. Außerdem wurde das Kopfbyte erweitert. Hier eine übersicht.

Das erweiterte Kopfbyte

Bit %7 aktiviert weiterhin die Standard-Dialogbox. Bit %0 bis %4 definieren auch unter MegaPatch den Schatten für eine Dialogbox.
Bit %6 definiert den Farb-Modus. Ist es gesetzt, dann sieht eine Dialogbox-Definition wie folgt aus:

```
* CODE *
:DlgBoxTab   b %01000001
             b $20,$7f
             w $0040,$00ff
             b Farbe
             ...
```

Nach den Größenangaben für die Dialogbox muß ein Byte für die Farbe der Dialogbox folgen. Dies funktioniert nur wenn es sich bei der Dialogbox nicht um eine Standardbox handelt.
Außerdem muß nun bei jedem Icon-Eintrag die Farbe für das Icon mit angegeben werden. Nachfolgend ein Beispiel für eine Icon-Definition:

```
* CODE *     b DBUSRICON
             b      $01,$10,Farbe
             b OK
             b      $01,$20,Farbe
             ...
```

Die Ergänzung durch Farbwerte ist nur dann erforderlich wenn man die Dialogbox immer mit bestimmten Farben versehen möchte. MegaPatch setzt Farbe für Dialogboxen ohne Bit %6 nach der vom Anwender definierten Farbtabelle (Siehe Tabelle 6: Die Farben des MegaPatch-Systems).

Wird Bit %5 auf #1 gesetzt, dann wird keine Farbe ausgegeben. Eine Dialogbox erscheint dann in den aktuellen Bildschirmfarben (abgelegt in :COLOR_MATRIX).

Bit %6 (Farbige Dialogbox) und Bit %5 (Farbe unterdrücken) schließen sich gegenseitig aus. Werden beide Bits auf #1 gesetzt, dann setzt MegaPatch unabhängig von der Kernal-Einstellung in :Flag_SetColor die Farbe wie in der Farbtabelle angegeben.
Diese Einstellung ist sinnvoll wenn ein Hintergrundbild verwendet wird und der Anwender im GEOS.Editor die Option "Dialogboxen in Farbe" deaktiviert hat: Hier würde ohne gesetzte Bits %6 und %5 die Dialogbox in den Farben des aktuellen Hintergrundbildes dargestellt werden.
Setzt man jedoch die Bits %6 und %5 auf #1, dann zeichnet MegaPatch in jedem Fall die Dialogbox in den aktuellen MegaPatch-Farben.

DRIVE ($07)

Dieser Steuercode existierte nur in früheren Versionen von GEOS-MegaPatch und stellte ein Symbol mit "LfWerk" dar. Damit konnte man zum jeweils nächsten Laufwerk wechseln. Wurde durch DBSETDRVICON ersetzt (siehe Beschreibung unter DBGETFILES=$0A)

DUMMY ($08)

Dieser Befehl wirkt wie der Assembler-Befehl NOP($EA) 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 einen neuen Dokuments. Diese könnte bisher wie folgt aussehen:

```
* CODE *
:DlgDefBox   b $81
             ...
             b DBUSRICON ,$0a,$48
             w ICON_ENTRY_LFWERK
             ...
             NULL
```

Hier kann man nun das Icon "Lfwerk" durch die vier Laufwerkicons ersetzen indem man DBUSRICON durch DRIVE ersetzt. Da für DRIVE kein WORD-Zeiger auf einen Iconeintrag benötigt wird, muß man das WORD "ICON_ENTRY_LFWERK" durch 2x DUMMY($08) ersetzen.
Außedem sollte man die X-Koordinate(Cards) korrigieren, da das "Lfwerk"-Icon 6 Cards breit ist, die vier Laufwerkicons jedoch bis zu 4x2 = 8 Cards in Anspruch nehmen können.

DBUSRFILES ($09)

Öffnet eine Dateiauswahlbox mit einer Liste von Dateien, welche vom Anwender frei definiert werden kann.

```
* CODE *    b DBUSRFILES
            w vecFNameTab
```

":vecFNameTab" zeigt dabei auf einen Speicherbereich mit max. 255 Dateinamen. Jeder Dateiname besteht aus 16 Zeichen und einem Null-Byte.
Ab Version V3.3r4 lässt sich DBUSRFILES mit DBSETDRVICON verknüpfen. Damit können die Laufwerk- Icons A: bis D: dargestellt werden.
Im Gegensatz zu DBGETFILES lässt sich DBUSRFILES nicht mit DBSELECTPART verknüpfen, da die beiden Modi den gleichen Speicherbereich verwenden.
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 Dateityp in :r7L.

DBSETCOL ($0A)

Erstellt ein Rechteck mit frei wählbarer Farbe.

```
* CODE *    b DBSETCOL, x1,y1,xb,yb,col
```

xl und yl sind als Bytewerte anzugeben und bezeichnen die linke, obere Ecke des Rechtecks. xb und yb geben die Breite bzw. die Höhe des Rechtecks an.
Für col kann ein Wert von 0 bis 15 für die Zeichenfarbe (High-Nibble) und auch für den Hintergrund (Low-Nibble) eingesetzt werden. Die Werte entsprechen den C64- Standardfarben.

DBGETFILES ($10)

Partitionsauswahlbox

Dieser Dialogbox-Steuercode wurde etwas erweitert. Setzt man hier Bit%7, z.B. über eine ODER-Verknüpfung, dann öffnet MegaPatch eine Partitionsauswahlbox. Eine Dialogboxtabelle sieht wie folgt aus:

```
* CODE *    b %10000001
            b DBGETFILES ! DBSELECTPART ,$00,$00
            b OK       , $00, $00
            b CANCEL, $00, $00
            b NULL
```

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

Laufwerk-Icons

Setzt man bei Verwendung von DBGETFILES das Bit#6, dann stellt MegaPatch für jedes verfügbare Laufwerk ein Laufwerk-Icon zur Verfügung. Möchte man also innerhalb der Auswahlbox auf ein anderes Laufwerk wechseln, so aktiviert man die Laufwerk-Icons wie folgt:

```
* CODE *
:DlgSlctFil b %10000001
            b DBGETFILES ! DBSETDRVICON ,$00,$00
            b OK     , $00, $00
            b CANCEL, $00, $00
            b NULL
```

Das Label :DBSETDRVICON ist mit %01000000 definiert. Innerhalb der Auswahlbox stehen jetzt die Icons "OK", "ABBRUCH" sowie für jedes verfügbare Laufwerk ein Laufwerk-Icon (A, B, C, D) zur Verfügung. Wenn der Anwender dann eines der Laufwerk-Icons anwählt, wird die Auswahlbox mit folgenden Werten in :sysDBData beendet:

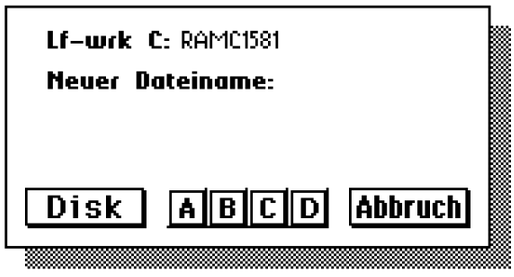
```
Laufwerk A:     $88
Laufwerk B:     $89
Laufwerk C:     $8A
Laufwerk D:     $8B
```

Die Routine zum Wechseln des Laufwerks könnte dann wie folgt aussehen:

```
* CODE *
:TestDlgJob lda   sysDBData
            bpl   :exit
            and   #%00001111
            jsr   SetDevice
            jsr   OpenDisk
            jmp   DoBoxAgain
```

Mit den hier beschriebenen Ergänzungen zur Dateiauswahlbox sollte jeder Programmierer eine Möglichkeit finden, spezielle Dialogboxen in eigenen Anwendungen zu vermeiden. Die neue Auswahlbox ist sehr flexiebel und sollte vielen Anforderungen gerecht werden.

Bild 1: Dialogbox mit Laufwerkicons...



(5) NEUE KERNAL-ROUTINEN

Erweiterte Sprungtabelle

Adresse ":i_UserColor" ($C0DC)

Parameter:	AKKU	Farbwert für Rechteck: Hintergrund Bits %0-%3 Zeichenfarbe Bits %4-%7
	Inlinedaten	Vier Byte-Werte in CARDs:
	xl	Linker Rand
	yo	Oberer Rand
	xb	Breite des Farbrechtecks
	yh	Höhe des Farbrechtecks
Rückgabe:	***	***
Verändert:	AKKU, X, Y Register :r5L, :r5H, :r6L, :r6H, :r7L, :r8	

Zeichnet ein Rechteck mit frei wählbarer Farbe. Die Koordinaten werden als Bytewerte direkt im Anschluß an den Aufruf erwartet. Die Farbe wird im AKKU übergeben. Beispiel:

```
* CODE *     lda   #$01
              jsr   i_UserColor
              b     x1,y1,xb,yb
```

Wie bei dem Dialogbox-Steuercode "DBSETCOL" steht hier xl und yl für die linke obere Ecke, xb und yb für die Breite bzw. die Höhe des Rechtecks. Die Farbe (hier $01=weiß) wird im Akku übergeben.

Adresse ":i_ColorBox" ($C0DF)

Parameter:	Inlinedaten	Fünf Byte-Werte inn Cards:
	xl	Linker Rand
	yo	Oberer Rand
	xb	Breite des Farbrechtecks
	yh	Höhe des Farbrechtecks
	col	Farbwert für Rechteck:
		Hintergrund Bits %0-%3
		Zeichenfarbe Bits %4-%7
Rückgabe:	***	***
Verändert:	AKKU, X, Y	
	Register :r5L, :r5H, :r6L, :r6H, :r7L, :r8	

Zeichnet ein Rechteck mit frei wählbarer Farbe. Die Koordinaten und der Farbwert werden als Bytewerte direkt im Anschluß an den Aufruf erwartet. Beispiel:

```
* CODE *     jsr    i_ColorBox
             b      xl,yo,xb,yb,col
```

Der Aufruf entspricht hier der Funktion DBSETCOL: xl und yl sind als Bytewerte anzugeben und bezeichnen die linke, obere Ecke des Rechtecks. xb und yb geben die Breite bzw. die Höhe des Rechtecks an.
Für col kann ein Wert von 0 bis 15 für die Zeichenfarbe (High-Nibble) und auch für den Hintergrund (Low-Nibble) eingesetzt werden. Die Werte entsprechen den C64-Standardfarben.

Adresse ":DirectColor" ($C0E2)

Parameter:	:r2L	Oberer Rand
	:r2H	Unterer Rand
	:r3	Linker Rand
	:r4	Rechter Rand
	Akku	Farbwert für Rechteck:
		Hintergrund Bits %0-%3
		Zeichenfarbe Bits %4-%7
Rückgabe:	***	***
Verändert:	AKKU, X, Y	
	Register :r5L, :r5H, :r6L, :r6H, :r7L, :r8	

Diese Routine erstellt ebenfalls ein Rechteck mit frei wählbarer Farbe. Dieser Aufruf erwartet die Koordinaten in den Registern :r2L, :r2H, :r3 und :r4. Die Belegung entspricht dabei der von Rectangle. Die Farbe wird im Akku übergeben. Beispiel:

```
* CODE *
:ClrColScrn LoadB r2L,$00
            LoadB r2H,$C7
            LoadW r3,$0000
            LoadW r4,$013F
            lda   #$02
            jsr   DirectColor
```

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

```
* CODE *
:ClrScreen  lda   #$02
            jsr   SetPattern
            jsr   i_Rectangle
            b     $00,$C7
            w     $0000,$013F
            lda   #$02
            jsr   DirectColor
```

Hier wird zuerst ein Rechteck mit dem Füllmuster $02 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.

Adresse ":RecColorBox" ($C0E5)

Parameter:	:r5L	Linker Rand
	:r5H	Oberer Rand
	:r6L	Breite des Rechtecks
	:r6H	Höhe des Rechtecks
	:r7L	Farbwert für Rechteck:
		Hintergrund Bits %0-%3
		Zeichenfarbe Bits %4-%7
Rückgabe:	***	***
Verändert:	AKKU, X, Y	
	Register :r5L, :r5H, :r6L, :r6H, :r7L, :r8	

RecColorBox stellt eine weitere Möglichkeit dar, ein Rechteck mit beliebiger Farbe zu zeichnen. Die übergabe der Farbdaten erfolgt hier jedoch immer in CARDs. Dir Register :r5L, :r5H, :r6L, :r6H und :r7L dienen zur übergabe der Parameter. Hier ein Beispiel:

```
* CODE *
:ClrScrCol  LoadB r5L,$00        ;Linker Rand
            LoadB r5H,$00        ;Oberer Rand
            LoadB r6L,$28        ;Breite
            LoadB r6H,$19        ;Höhe
            LoadB r7L,$02        ;Farbwert
            jsr   RecColorBox
```

Die Verwendung von Bytewerten kann den Programmcode um ein paar zusätzliche Bytes reduzieren. Ein Anwendungsbeispiel wäre die Farbwerte aus einer Tabelle mittels einer Kopierschleife nach :r5L bis :r7L zu kopieren und dann RecColorBox aufzurufen.

Adresse ":GetBackScreen" ($C0E8)

Parameter:	keine	
Rückgabe:	***	***
Verändert:	AKKU, X, Y	
	Register :r0 bis :r8, :r11	

Kopiert die Hintergrundgrafik aus dem erweiterten RAM in den Vordergrundbildschirm des C64. Entscheidend hierfür ist das im Register :sysRAMFlg das Bit %3 auf #1 gesetzt ist, dann wird die Hintergrundgrafik aus dem RAM eingelesen.
Ist Bit %3 auf #0 gesetzt, dann löscht diese Routine den Bildschirm mit einem vordefinierten Muster (enthalten im Register :BackScrPattern).

Adresse ":ResetScreen" ($C0EB)

Parameter:	keine	
Rückgabe:	***	***
Verändert:	AKKU, X, Y Register :r0 bis :r8, :r11	

Diese Routine ist Teil der :InitGEOS-Routine und löscht den Bildschirm und aktiviert die Standardfarben für Hintergrund und Rahmen. Möchte man schnell den Bildschirm löschen (Füllmuster $02), dann kann diese Routine verwendet werden.

Adresse ":GEOS_InitSystem" ($C0EE)

Parameter:	keine	
Rückgabe:	***	***
Verändert:	AKKU, X, Y Register :r0, :r1, :r2L	

Setzt alle GEOS-Register auf Standardwerte zurück. Kann innerhalb einer Anwendung verwendet werden, wenn z.B. ein anderes Programm gestartet werden soll oder innerhalb eines Menüs um alle Systemvektoren vor dem Aufruf eines Menüs zu löschen.

Adresse ":PutKeyInBuffer" ($C0F1)

Parameter:	AKKU	Tastaturcode
Rückgabe:	***	***
Verändert:	AKKU, X Register :pressFlag	

Der C64-BASIC-Editor besitzt die Möglichkeit des "programmierten Direktmodus". Hierbei werden in den Tastaturpuffer Tastencodes eingetragen, welche dann später ausgeführt werden.
Diese Routine kopiert einen Tastencode in den Tastaturpuffer welcher dann nach :keyData kopiert wird. Ein Anwendungsbespiel wäre z.B. bei einer Tastatureingabe über GetString den Vektor :otherPressVec auf folgende Routine zu setzen:

```
* CODE *
:ExGetStrg  lda   #CR
            jmp   PutKeyInBuffer
```

Wenn der Anwender jetzt einen MouseButton drückt, dann wird diese Routine ausgeführt und der Tastencode <CR> = RETURN in den Tastaturpuffer geschrieben. Beim nächsten Durchlauf der Mainloop wird dann die Tastatureingabe beendet.

HINWEIS: Dies funktioniert nur mit dem Original GEOS-MegaPatch. Wird MegaPatch für die PC-Tastatur "geoKeys" angepaßt, dann passiert bei obigen Beispiel nichts, da der PC-Tastatur-Treiber Tasteneingaben auf andere Art und Weise verwaltet!

Adresse ":SCPU_Pause" ($C0F4)

Parameter:	keine	
Rückgabe:	***	***
Verändert:	AKKU	

Häufig ist es notwendig den C64 eine genau definierte Zeit "warten" zu lassen. übliche "LDX_DEX_BNE_LOOP"-Schleifen können nicht immer verwendet werden, da Sie den Nachteil haben, auf unterschiedlichen Systemen unterschiedlich schnell abgearbeitet zu werden: Auf einem C128 ist eine solche Schleife doppelt so schnell wie auf einem C64, auf einem SCPU64-System sogar bis zu 20x schneller.

SCPU_Pause wartet hier immer circa eine 1/10-Sekunde.

Adresse ":SCPU_OptOn" ($C0F7)

Parameter:	keine	
Rückgabe:	***	***
Verändert:	AKKU, X, Y	

Aktiviert die Optimierung auf SuperCPU-Systemen für MegaPatch. Im Normalfall ist die Optimierung aktiviert. Ist keine SuperCPU vorhanden bleibt dieser Aufruf ohne Funktion.

Adresse ":SCPU_OptOff" ($C0FA)

Parameter:	keine	
Rückgabe:	***	***
Verändert:	AKKU, X, Y	

Deaktiviert die Optimierung für die SuperCPU. Dies kann notwendig werden, wenn z.B. in den Textmodus geschaltet werden soll (wie etwa beim Programm geoDebugger oder geoBasic). Ist keine SuperCPU vorhanden bleibt dieser Aufruf ohne Funktion.

Adresse ":SCPU_SetOpt" ($C0FD)

Parameter:	Flag_Optimize	$00 = Optimierung ein $03 = Optimierung aus
Rückgabe:	***	***
Verändert:	AKKU, X, Y	

Setzt oder löscht die Optimierung für die SuperCPU. Dazu muß im Register :Flag_Optimize der Wert $00 eingetragen werden wenn die Optimierung aktiviert oder $03 wenn die Optimierung abgeschaltet werden soll. Ist keine SuperCPU vorhanden bleibt dieser Aufruf ohne Funktion.

(6) DIE AUSGELAGERTEN KERNAL-ROUTINEN

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

Programmierer müssen auf diese Routinen normalerweise nicht zurückgreifen. Ausnahme bildet das Registermenü, dieses wird im normalen Anwenderspeicher ausgeführt. Programmierer, die Programme in diesem Bereich ausführen, müssen das beachten! Die Ladeadresse kann man der folgenden Tabelle entnehmen:

Tabelle 7: Erweiterte Routinen

Routine	Start	Ende	Größe	Start in DACC
Register 1)	$6D00	$78FF	$0C00	$0000
EnterDeskTop 2)	$7E00	$7FFF	$0200	$0C00
Panic	$8000	$80FF	$0100	$0E00
ToBASIC	$8E00	$8FFF	$0200	$0F00
GetNxDay	$8000	$807F	$0080	$1100
DoAlarm	$8000	$807F	$0080	$1180
GetFiles 2)	$6000	$7BFF	$1C00	$1200
GetFiles_Data 2) 3)	dlgBoxRamBuf		$0180	$2E00
GetFiles_Menu 2)	$8000	$837F	$0380	$2F80
DB_SCREEN	$8000	$82FF	$0300	$3300
DB_COLOR			$03E8	$3600
DB_GRAFX			$1F40	$39E8
GetBackScreen	$8000	$80FF	$0100	$5928
BS_COLOR			$03E8	$5A28
BS_GRAFX			$1F40	$5E10
ScreenSaver 1) 2)	$6400	$7FFF	$1C00	$7D50
SS_COLOR			$03E8	$9950
SS_GRAFX			$1F40	$9d38
Spooler 2)	$4000	$55FF	$1600	$BC78
Infobock Spooler 5)			$0100	$D180
Spooler/Druckertreiber 4)	$7900	$7F3F	$0640	$D280
InfoBlock Druckerttreiber			$0100	$D8C0
Druckertreiber 4)	$7900	$7F3F	$0640	$D9C0
TaskManager	$4000	$5FFF	$2000	$E000

Anmerkungen zu Tabelle 7:

1) Diese Routinen können auch vom Anwender aufgerufen werden. Man sollte jedoch darauf achten, das eigene Programme nicht die Startadresse der Routine überschreiben. Werden die Routinen nicht benötigt, kann der komplette Anwendungsspeicher (APP_RAM=$0400 bis OS_VARS=$7FFF) für Anwendungen genutzt werden.

2) Diese Routinen werden aus dem Kernal heraus aufgerufen. Während diese Routinen ausgeführt werden, wird der Speicherinhalt in der Speichererweiterung ausgelagert. Eigene Programme müssen deshalb keinerlei Rücksicht auf diese Routinen und deren Speicherlage nehmen.

3) Diese Routinen dürfen nicht im normalen Anwendungspeicher ausgeführt werden, da diese sonst Kernal-Funktionen oder GEOS-Anwendungen negativ beeinflussen könnte. Als Startadresse wurde daher eine Adresse gewählt, welche im Normalfall dem GEOS-Kernal vorbehalten ist.

4) Bedingt durch einen Fehler im Programm "geoCalc64" wurde in der MegaPatch-Version von 2018 die Größe von "Druckertreiber im RAM" und "Spooler/Druckertreiber" um 1 Byte reduziert. über den GEOS.Editor kann die Größe eingestellt werden, siehe GEOS.Editor/Drucker/GeoCalc-Fix.

5) Reserviert für den Infoblock des Spooler/Druckertreibers. Aktuell für künftige Erweiterungen in MegaPatch reserviert, da beim laden des Druckertreibers über eine Anwendung der Infblock des eigentlichen Druckertreibers nach :fileHeader geladen wird.

Das Registermenü

Das Registermenü stellt eine Vielzahl neuer Funktionen und Routinen zur Verfügung. Dazu übergibt man in :r0 einen Zeiger auf eine Register-Tabelle ähnlich :DoMenu Vorher muß jedoch die Registermenü-Routine in den Speicher geholt werden. Dazu kann man folgende Routine verwenden:

```
* CODE *
:LdRegMenu   jsr    SetADDR_Register
             jsr    FetchRAM
```

Hier nun der grundlegende Aufbau eines Registermenüs:

```
* CODE *
:RMenuTab    b yoben, yunten
             w xlinks, xrechts
             b Anzahl Register
             w RNmIcon01
             w RTabDat01
             w RNmIcon02
             w RTabDat02
             w RNmIcono3
             w RTabDat03
             ...
:RNmIcon01   w RRIcon01
             b xpos,ypos,breite,hoehe
:RNmIcon02   w RRIcon02
             b xpos,ypos,breite,hoehe
:RNmIcon03   w RRIcon03
             b xpos,ypos,breite,hoehe

:RTabDat01   b Anzahl Eintraege
             b Typ Register-Eintrag
             (10 Byte Systemdaten)
             b Typ Register-Eintrag
             (10 Byte Systemdaten)
             ...
```

Die ersten sechs Bytes bestimmen die Größe der Registerkarten. Es ist zu beachten das die Register- Kartenreiter hier nicht enthalten sind. Die größte Ausdehnung für ein Registermenü wäre demnach:

```
* CODE *
:RMenTab     b $08,$c7
             w $0000,$013F
```

Danach folgt die Anzahl der Registerkarten. Zu der Anzahl der Register muß nun die passende Anzahl Registerverweise 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 Registerseite.

Der Eintrag für das Kartenreiter-Icon ähnelt einem Eintrag in der :Dolcons-Tabelle. Als erstes wird ein 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 ebenfalls in CARDs angegeben, die Höhe des Icons wiederum in Pixel.

Für ein Registermenü stehen verschiedene Befehle zur Verfügung um die Registerseite zu gestalten. Hier die übersicht über die einzelnen Befehlsbytes:

Befehlskennbytes

BOX_USER ($01)

Definiert einen Bereich auf dem Bildschirm der mit der Maus angeklickt werden kann. Im Anschluß wird dann eine frei definierbare Routine aufgerufen die den Mausklick auswerten kann. Beispiel:

```
* CODE *     b BOX_USER
             w BOX_NAME
             w BOX_ROUTINE
             b yoben, yunten
             w xlinks, xrechts
```

BOX_NAME zeigt auf einen Text der die Box auf dem Bildschirm beschreibt. Da der Text nicht an eine bestimmte Stelle festgelegt werden kann, muß der Text mit einem WORD für die X-Koordinate und einem BYTE für die Y-Koordinate beginnen. Soll kein Text ausgegeben werden ist BOX_NAME=$0000.

BOX_ROUTINE zeigt auf die Anwender-Routine. Diese wird zum einen während dem Aufbau des Registermenüs aufgerufen und zum anderen wenn der Anwender den Bereich mit der Maus angeklickt hat.
Um nun unterscheiden zu können wann die Routine aufgerufen wurde, kann man das Register :r1L abfragen: Hat es den Wert $00, dann wird das Register aufgebaut, ist es $FF dann wurde der Bereich mit der Maus angeklickt und das Programm soll den Mausklick auswerten. Die Koordinaten yoben, yunten, xlinks und xrechts bestimmen die Größe.

BOX_USER_VIEW ($02)

Dieser Befehl entspricht BOX_USER mit einer Ausnahme: Die Anwenderroutine kann nur mit dem Registerwert :r1L = $00 aufgerufen werden. Ein anklicken mit der Maus ist nicht möglich.
Ein "BOX_USER_VIEW"-Befehl kann z.B. dazu verwendet werden einen Parameter kurzfristig zu deaktivieren. Dazu ersetzt man in der Registertabelle den "BOX_USER"-Code durch "BOX_USER_VIEW".

BOX_USEROPT ($03)

In der Funktion identisch mit "BOX_USER". Hier wird jedoch vor Aufruf der Anwenderroutine ein Eingabefeld in der Größe der angegebenen Koordinaten auf die Registerkarte gezeichnet.

BOX_USEROPT_VIEW ($04)

Wie "BOX_USER_VIEW" erlaubt auch dieser Befehl nur die Darstellung von Informationen während dem Aufbau des Registers auf dem Bildschirm. Eine Abfrage mit der Maus ist nicht möglich.

BOX_FRAME ($05)

Dieser Befehl dient zum Einteilen der Registerkarte in verschiedene Bereiche. Nach dem Befehlsbyte folgt ein WORD als Zeiger auf den Frame-Titel. Dieser wird nicht mit einer Koordinatenangabe begonnen, da der Titel immer am oberen Rand des Frames dargestellt wird.

BOX_ICON ($06)

Stellt ein Icon auf der Registerkarte dar. Der benötigte Definitionsblock für ein Icon sieht wie folgt aus:

```
* CODE *
:RTabDat01  b $01                ;Anz. Funktionen
            b BOX_ICON
            w BOX_TEXT
            w BOX_ROUTINE
            b ypos               ;In Pixel!
            w xpos
            w RegTDatIcon1       ;Icon-Eintrag
            b BOX OPT
```

"BOX_TEXT" ist wieder ein Zeiger auf einen Text der wie bei "BOX_USER"/"BOX_USEROPT" mit einer Koordinatenangabe beginnt. "BOX_ROUTINE" ist ein Zeiger auf eine Routine die ausgeführt wird, wenn das Icon mit der Maus angeklickt wurde. ypos und xpos beziehen sich auf die linke obere Ecke des Icons. Zu beachten ist das die X-Koordinate hier in Pixel angegeben wird. Das nächste WORD zeigt auf die Icon-Definitionstabelle wie sie auch bei :DoIcons verwendet wird:

```
* CODE *
:RTDatIcon1 w Icon01
            b xpos,ypos,breite,höhe
```

xpos und breite sind in CARDs anzugeben, ypos und höhe in Pixel.

Der letzte Wert BOX_OPT im Definitionsblock zeigt auf einen Eintrag in der Registertabelle der aktualisiert werden muß wenn die Icon-Routine ausgeführt wurde.

BOX_ICON_VIEW ($07)

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

BOX_OPTION ($08)

Häufig ist es notwendig bestimmte Funktionen innerhalb eines Programms ein- oder auszuschalten. Dafür bietet dieser Befehl eine einfache Möglichkeit. Dazu wird bei jedem "BOX_OPTION"-Befehl ein kleines Eingabefeld auf den Bildschirm gebracht, welches der Anwender mit der Maus anklicken kann. Eine aktivierte Option wird dann mit einem Haken markiert. Hier nun ein Beispiel:

```
* CODE *
:RTabDat02  b $01
            b BOX_OPTION          ;Befehlsbyte
            w BOX_TEXT            ;Beschreibung
            w BOX_ROUTINE         ;Anwender-Routine
            b ypos                ;Koordinatenangabe
            w xpos
            w VECTOR              ;Zeiger auf Byte
            b BIT_FLAG            ;Bit-Maske
```

BOX_TEXT zeigt wieder auf eine Beschreibung die mit einer Koordinaten-Angabe beginnen muß. BOX_ROUTINE zeigt auf eine eigene Routine, welche ausgeführt wird, wenn die Option ausgewählt wurde. VECTOR zeigt auf ein Datenbyte. Welche Bits innerhalb dieses Bytes geändert werden gibt das nächste Byte BIT_FLAG an.

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

Zuerst ändert die Registerroutine die Bits im Datenbyte welche in BIT_FLAG mit %1 markiert wurden. Dazu wird ein einfacher EOR-Befehl verwendet. Danach wird BOX-ROUTINE aufgerufen. Hier kann der Anwender dann feststellen ob das ändern der Option überhaupt zulässig war. Falls nicht kann man hier die Bits wieder zurücksetzen. Im Anschluß prüft die Registerroutine ob die BIT_FLAG-Bits im Datenbyte gesetzt sind. Wenn ja, wird die Option als "Aktiv" 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. Ein Beispiel dafür ist die Anzeige im GEOS.Editor ob eine SuperCPU aktiviert ist oder nicht. Da man die SuperCPU im laufenden Betrieb nicht deaktivieren kann, ist ein ändern der Option auch nicht möglich.

BOX_STRING ($0A)

Gibt einen String auf dem Bildschirm aus. Dazu wird zuerst ein Eingabefeld auf dem Bildschirm dargestellt und dann der Text ausgegeben. Dieser kann vom Anwender geändert werden, indem das
Eingabefeld mit der Maus angeklickt wird.

```
* CODE *
:RTabDat03  b $01
            b BOX_STRING          ;Befehlsbyte
            w BOX_TEXT            ;Beschreibung
            w BOX ROUTINE         ;Anwender-Routine
            b ypos                ;Koordinatenangabe
            w xpos
            w STRING_VECTOR       ;Zeiger auf Text
            b ANZAHL ZEICHEN      ;Textlänge
```

BOX_TEXT, BOX_ROUTINE, ypos und xpos wurden bereits beschrieben.
STRING_VECTOR zeigt auf einen Textstring der während des Aufbaus des Registermenüs auf dem Bildschirm ausgegeben wird. Soll zu Beginn kein Text ausgegeben werden, muß der String mit einem Null-Byte beginnen. Die Länge des Eingabefeldes wird mit dem Byte ANZAHL_ZEICHEN festgelegt.

BOX_STRING_VIEW ($0B)

Wie BOX_STRING stellt dieser Befehl ein Eingabefeld mit einem Textstring auf dem Bildschirm dar. Dieser Text kann vom Anwender jedoch nicht modifiziert werden.

BOX_NUMERIC ($0C)

Um die Eingabe von Zahlen zu vereinfachen, wurde BOX_NUMERIC entwickelt. Auch hier wird wie bei BOX_STRING ein Eingabefeld auf dem Bildschirm dargestellt. Um das Aussehen der Zahl zu bestimmen, kann die Variable ANZAHL mit verschiedenen Parametern verknpüft werden:

Tabelle 8: Parameter für Zahlenausgabe

Parameter	Wert	Beschreibung
NUMERIC_LEFT	$00	Zahl im Eingabefeld linksbündig
NUMERIC_RIGHT	$80	Zahl im Eingabefeld rechtsbündig
NUMERIC_SETSPC	$00	Führende Leerstellen mit ' '-Zeichen füllen
NUMERIC_SET0	$40	Führende Leerstellen mit '0'-Zeichen füllen
NUMERIC_BYTE	$00	Die Zahl ist ein Byte
NUMERIC_WORD	$20	Die Zahl ist ein WORD im LOW/HIGH-Format

Eine typische Tabelle könnte wie folgt aussehen:

```
* CODE *
:RTabDat01  b $01
            b BOX_NUMERIC          ;Befehlsbyte
            w BOX_TEXT             ;Beschreibung
            w BOX_ROUTINE          ;Anwender-Routine
            b ypos                 ;Koordinatenangabe
            w xpos
            w VECTOR               ;Zeiger Byte/Word
            b $04 ! NUMERIC WORD ! NUMERIC RIGHT
```

VECTOR zeigt auf eine Speicherstelle mit dem zu ändernden Byte/Word. In diesem Beispiel kann das WORD max. vier Zeichen lang sein was Zahlen im Bereich von 0-9999 zuläßt. Eine detaillierte Wertabfrage kann über BOX_ROUTINE erfolgen. Ist der Wert ungültig kann hier der Wert noch korrigiert werden bevor die Registerroutine 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 verändern zu lassen.

Registermenü - Variablenübersicht

Anbei nun eine Liste der zusätzlichen Variablen, welche das Register-Menü zur Verfügung stellt. Zuerst die neuen Variablen:

```
: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
```

Einsprungtabelle

Hier nun die Sprungtabelle am Anfang der Register-Routine.

```
:DoRegister             = $6d00   ;Menü aktivieren
:ExitRegisterMenu       = $6d03   ;Menü deaktivieren
:RegisterInitMenu       = $6d06   ;Zeichnet das aktuelle Menü neu
:RegisterUpdate         = $6d09   ;Aktualisiert die aktuelle Option
:RegisterAllOpt         = $6d0C   ;Zeichnet alle Karten neu
:RegisterNextOpt        = $6d0f   ;Zeichnet aktuelle Karte neu
:RegDrawOptFrame        = $6d12   ;Zeichnet Rahmen um Eingabefeld
:RegClrOptFrame         = $6d15   ;Löscht Rahmen um Eingabefeld
:RegisterSetFont        = $6d18   ;Aktiviert Register-Zeichensatz
:RegisterAktiv          = $6d1b   ;Byte: Enthält aktives Register
```

Adresse ":DoRegister" ($6D00)

Parameter:	:r0	Zeiger auf Register-Tabelle
Rückgabe:	***	***
Verändert:	AKKU, X, Y Register :r0 bis :r15	

Diese Routine installiert das Register-Menü und stellt das Menü auch auf dem Bildschirm dar. Wer zusätzlich Icons über :DoMenu oder auch PullDown-Menüs über :DoMenu verwenden möchte, der kann das vorher tun. :DoRegister kann parallel dazu installiert werden. Außer dem Vector in :r0 sind keine weiteren Parameter erforderlich.
Da beim Aufbau des Registermenüs auch Anwender-Routinen ausgeführt werden, können praktisch alle Register- und Variablen verändert werden. Generell sollte man das Register :r15 während dem Aufbau des Registermenüs nicht ändern, da hier der Zeiger auf die aktuelle Registertabelle abgelegt wird.

Adresse ":ExitRegisterMenu" ($6D03)

Parameter:	keine	
Rückgabe:	***	***
Verändert:	AKKU, X, Y Register :r0 bis :r15	

Die Register-Routine liegt im Bereich des Anwenderspeichers. Bevor man wieder auf den gesammten Speicher zugreifen kann sollte man diese Routine aufrufen. Damit werden Vektoren für Zeichensatz und GEOS-Vektoren wie :otherPressVec wieder zurückgesetzt.

Adresse ":RegisterInitMenu" ($6D06)

Parameter:	keine	
Rückgabe:	***	***
Verändert:	AKKU, X, Y Register :r0 bis :r15	

Falls es notwenig wird, das aktuelle Register neu zu zeichnen, so verwendet man diese Routine. Danach werden alle Einträge auf der aktuellen Seite neu gezeichnet. Dies kann unter Umständen dann notwendig sein, wenn sich einige Optionen gegenseitig ausschließen.

Adresse ":EnterDeskTop" ($C22C)

Diese Routine versucht den aktuellen DeskTop zu starten. Dabei sucht MegaPatch zuerst auf RAM-Laufwerken nach einer Kopie des DeskTop.
Wird hier kein DeskTop gefunden, so wird die Suche auf Diskettenlaufwerken fortgesetzt.
Wenn auch hier kein DeskTop gefunden wird, so erscheint die Meldung:

"Bitte Diskette mit DeskTop einlegen!"

Die dazugehörige Dialogbox befindet sich auch weiterhin im Kernal, da einige DeskTop-Varianten diese verändern.
Um ein höchstmaß an Kompatibilität zu erreichen, sollte innerhalb des Kernals nichts verändert werden. Auch der Name der eigenen DeskTop-Variante muß im Kernal nicht geändert werden. Einfacher ist das umbenennen der eigenen Oberfläche in "DESK TOP". So ist garantiert das immer ein DeskTop gefunden wird.

Wer einen eigenen DeskTop entwickelt, der nicht auf Diskette gespeichert sondern im erweiterten RAM abgelegt werden kann, der muß hier eine eigene Startroutine installieren. Diese muß zuerst in den erweiterten MegaPatch-Speicher übertragen werden.

```
* CODE *
:InstOwnDT   jsr    SetADDR_EnterDT
             LoadW r0,OwnLoader
             jsr    StashRAM
             ...
```

Die neue DeskTop-Laderoutine befindet sich nun im GEOS-DACC und wird immer dann aufgerufen, wenn eine Applikation beendet wurde. Eine solche RAM-Startroutine könnte wie folgt aussehen:

```
* CODE *
:OwnLoader   sei
             cld
             ldx    #$ff            ;Wichtig für
             stx    firstBoot       ;AutoBoot-Vorgang.
             txs

;Variablen und Bildschirm löschen.
             jsr    GEOS_InitSystem
             jsr    ResetScreen

:FetchOwnDT  LoadW r0,APP_RAM      ;APP_RAM = $0400
             LoadW r1,$0000
             LoadW r2,$4000
             lda    #$01            ;RAM-Bank in der
             sta    r3L             ;sich der eigene
             jsr    FetchRAM        ;DeskTop befindet.

             LoadB r0L,%00000000 ;DeskTop starten.
             LoadW r7,APP_MAIN
             jmp    StartAppl
```

Ab :OwnLoader steht die Standard-Initialisierung für GEOS, diese sollte in jedem DeskTop-Lader enthalten sein. GEOS_InitSystem und ResetScreen sind Teil der neuen Sprungtabelle, so daß keine Anpassung an spezielle MegaPatch-Versionen erforderlich ist. Ab :FetchOwnDT steht die Routine, welche den eigenen DeskTop startet.

Die obige Routine ist im Speicher frei verschiebar. Wer sichergehen will, das seine Routine im richtigen Speicherbereich abläuft, der sollte mittels SetADDR_EnterDT($CFE3) das Register :r0 prüfen. An diese Adresse wird die EnterDeskTop-Routine in den Computerspeicher geladen und ausgeführt.
Man sollte sich jedoch nicht auf eine Adresse festlegen, da spätere MegaPatch-Versionen die Routine evtl. in einen anderen Speicherbereich laden können.

Adresse ":ToBasic" ($C241)

Diese Routine ist komplett aus GEOS-V2 übernommen worden.
ACHTUNG! Wer auf der RAMLink ein AutoStart-Menü eingerichtet hat, der kann keine Programme unter GEOS starten. Nach einem Reset wird automatisch das RAMLink-AutoBoot-Menü aktiviert.

Adresse ":PANIC" ($C2C2)

Jeder Programmierer kennt die Panic!-Dialogbox, mancher Anwender leider auch. Diese Routine erscheint wenn innerhalb von einer Applikation ein ungültiger Befehl auftaucht ("Absturz nahe $XXXX"). Die Panicbox fängt leichte Fehler ab und kehrt danach zum DeskTop zurück. Bei schweren Fehlern (illegale Befehle) hilft auch diese Routine nicht mehr.

Adresse ":GetNxDay" (intern)

Diese Routine ist relativ uninteressant für den Programmierer. Sie wird alle 24 Stunden von GEOS aufgerufen, wenn die GEOS-Uhr einen neuen Tag anzeigt.

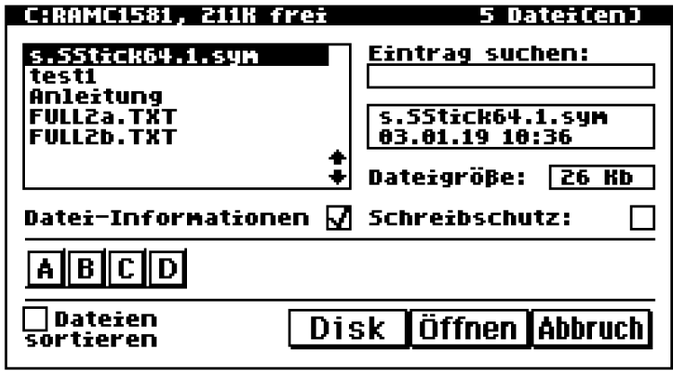
Adresse ":DoAlarm" (intern)

Wird mit dem GEOS-Wecker eine Weckzeit eingestellt, so wird vom GEOS-Kernal nach erreichen der Alarmzeit diese Routine ausgeführt. Im Gegensatz zur Originalroutine zeigt MegaPatch den Alarm akustisch und optisch an.

Adresse ":GetFiles" (intern)

Die neue Dateiauswahlbox des GEOS-MegaPatch. Diese Routine wird im Speicer ab $6000 ausgeführt. Beispiel für eine Dateiauswahlbox:

Bild 2: Dateiauswahlbox mit Laufwerk-Icons...



Hier sieht man die neue Dateiauswahlbox unter GeoWrite. Im Gegensatz zur alten Dateiauswahlbox stellt diese bis zu 255 Dateien im Fenster dar. Mittels der Rollpfeile und des Rollbalkens kann man sich in der Liste bewegen.

Außerdem ein Feature von MegaPatch das Programmierern viel Aufwand abnehmen kann: Die vier Laufwerk-Icons des MegaPatch-Systems. Diese können mit einem Steuerbefehl inn der DialogBox aktiviert werden. Siehe dazu (1)Geänderte Kernal-Funktionen.

Die notwendigen Patches für GeoWrite (und evtl. auch andere Standard-Applikationen) werden von anderen Programmierern zur Verfügung gestellt (z.B. auch das Öffnen von Dokumenten von allen vier Laufwerken über Dateiauswahlbox und über Doppelklick vom DeskTop aus).

Wer eine bestimmte Datei suchen möchte, der kann im Feld "Eintrag suchen" einzelne Zeichen eingeben. Nach <RETURN> sucht die Auswahlbox nach der ersten Datei die mit den eingegebenen Zeichen beginnt. Jeder weitere Druck auf <RETURN> zeigt die nächste Datei an. Findet die Auswahlbox einen Eintrag der genau der eingegebenen Zeichenfolge entspricht wird die Datei automatisch geöffnet. Das Eingabefeld beachtet die Groß/Kleinschreibung!

Unterhalb des "Eintrag suchen"-Feldes wird der Name der gerade markierten Datei angezeigt. Außerdem noch Datum und Uhrzeit der letzten Änderung. Darunter steht die aktuelle Dateigröße und der Schreibschutz-Zustand. Dieser kann mit einem Mausklick geändert werden.

Am unteren Rand der Dialogbox steht die Option "Dateien sortieren" zur Verfügung. Mit einem Klick wird die Liste der Dateien alphabetisch sortiert. Dies funktioniert nicht, wenn die Dateiauswahlbox über den Dialogboxcode DBUSRFILES aufgerufen wurde. Hier könnte die Dateitabelle mit anderen Tabellen verknüpft sein was im schlimmsten Fall zur falschen Auswertung eines Eintrages führen kann.

Adresse ":GetFiles_Data" (intern)

Diese 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.
Da bei DBUSRFILES praktisch der komplette Speicher von :APP_RAM ($0400) bis :APP_VAR ($7F40) als Ablagebereich für Tabelleneinträge dienen kann, wird diese Routine ab :dlgBoxRamBuf ausgeführt.
Für den Anwender hat diese Routine keinerlei Bedeutung.

Adresse ":GetFiles_Menu" (intern)

Auch diese Routine wird von GetFiles aufgerufen und zeichnet die Auswahlbox mit allen Icons auf den Bildschirm. Diese Routine wird außerhalb der eigentlichen GetFiles-Routine ausgeführt da es sonst zu Problemen beim DoIcons-Aufruf kommen kann. Würde die Dateiauswahlbox aus einem Programm heraus gestartet, welches ab $6000 im Speicher liegt, dann könnte DoIcons die selbstdefinierten Icons nicht korrekt darstellen.
Der Grund dafür ist das Programm und die Dialogboxtabelle sich zu diesem Zeitpunkt im ausgelagerten RAM befinden.
HNWEIS: Bei GateWay führt dies allerding trotzdem zu einem Problem. Wird die Datei-Info aufgerufen, dann wird in der Dialogbox auch das Datei-Icon angezeigt. Das Datei-Icon liegt aber genau im Bereich in dem diese Routine ausgeführt wird (:fileHeader = $8100). Das Datei-Icon wird daher nicht korrekt angezeigt.
Für den Anwender hat diese Routine keinerlei Bedeutung.

Adresse ":TaskManager" (intern)

Der TaskManager erlaubt es bis zu neun Anwendungen gleichzeitig geöffnet zu halten. Mittels der Tastenkombination <CBM> und <CONTROL> 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 MegaPatch nach $4000 in den Speicher geladen und dort ausgeführt. Die ersten drei Bytes beinhalten einen Sprungbefehl zum Hauptprogramm. Danach folgt eine Byte-Tabelle:

```
: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 die reservierten 64K-Speicherbänke, eine Speicherbank für jeden Task. Nicht installierte Tasks enthalten hier den Wert $00.

```
: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, das der entsprechende Task durch eine geöffnete Anwendung belegt ist. Wird eine Applikation durch den Befehl "jmp EnterDeskTop" beendet, so wird hier der Inhalt der entsprechenden Variable gelöscht und der TaskSpeicher steht wieder zur Verfügung.

```
:MaxTaskInstalled       b $09
```

Max. Anzahl installierter Tasks. Die MegaPatch-Demo-Version setzte hier den Wert $02 (max. Anzahl Tasks ist war auf zwei Anwendungen beschränkt). Seit 2018 gibt es diese Beschränkung nicht mehr.

```
: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 die im entsprechenden Task zuerst geöffnet wurde. Wird innerhalb der laufenden Anwendung eine andere Anwendung geöffnet, bleibt der Name der ersten Anwendung hier erhalten.

Adresse ":DB_SCREEN" (intern)

Die neue Dialogbox des GEOS-MegaPatch rettet den Bildschirminhalt vor Aufbau der Dialogbox in einen reservierten Speicherbereich. 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
               jsr    DB_SCREEN_SAVE

;Bildschirm-Inhalt zurücksetzen.
               jsr    SetADDR_DB_SCRN
               jsr    DB_SCREEN_LOAD
```

Vor dem Aufruf muß man lediglich in $003F/$0040 einen Zeiger auf Dialogboxtabelle einrichten. Da dieses aber vom GEOS-Kernal erledigt wird, hat diese Routine keine Bedeutung für Programmierer.

Adresse ":GetBackScreen" ($C0E8)

Diese Routine lädt das Hintergrundbild und stellt es im Vordergrundbildschirm dar. Ist kein Hintergrundbild definiert, dann wird der Bildschirm mit dem Füllmuster in :BackScrPattern gefüllt.
Für diese Routine existiert ein Sprungbefehl in der neuen Sprungtabelle. Siehe dazu (3) Neue Kernal-Funktionen.

Adresse ":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:

```
* CODE *
;Bildschirmschoner starten.
:MainInit    jmp   InitScreenSaver

;Bildschirmschoner initialisieren.
:InstallSvr  jmp   InstallScrSaver

:SaverName   b "Starfield" ,NULL

:InitScrSvr  php
             sei
             ldx   #$1f            ;ZeroPage speichern.
::51         lda   r0L,x
             pha
             dex
             bpl   :51
             jsr   DoSaverJob      ;Effekt starten.
             lda   #%01000000      ;ScreenSaver-Reset.
             sta   Flag_ScrSaver
             ldx   #$00            ;ZeroPage laden.
::52         pla
             sta   r0L,x
             inx
             cpx   #$20
             bne   :52
             plp
             rts
```

Diese Routine sollte in jedem Bildschirmschoner enthalten sein. Wichtig ist das die ersten Bytes 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:

```
* CODE *
: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
            sta   r3L
            jsr   StashRAM
```

Bevor man den Bildschirmschoner beendet, muß man den Bildschirminhalt wiederherstellen. Dazu kann man die folgende Routine verwenden:

```
* CODE *
: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
            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:

```
* CODE *     lda   #$00
             sta   $dc00
             lda   $dc01
             eor   #$ff
             bne   EndScreenSaver
```

Man sollte auf eine Abfrage der Mausbewegung verzichten, da bei installierter SuperCPU die Maus leicht "zittert" was zur sofortigen Beendigung des Bildschirmschoners führen würde.

Sprungtabelle für ausgelagerte Kernal-Routinen

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. Wer die erweiterten Kernal-Routinen ändern möchte kann dies tun, man sollte jedoch darauf achten, das die neue Größe die im Register :r2 enthaltene Größe nicht überschreitet.

Tabelle 9: Sprungtabelle für ausgelagerte Kernal-Routinen

Routine	Adr.	Beschreibung
SetADDR_TaskMan	$CFED	TaskManager-Routine
SetADDR_Register	$CFE6	Registermenü-Routine
SetADDR_EnterDT	$CFE3	EnterDeskTop-Routine
SetADDR_ToBASIC	$CFE0	Routine zum beenden von GEOS
SetADDR_PANIC	$CFDD	PANIC!-Dialogbox
SetADDR_GetNxDay	$CFDA	Datumswechsel
SetADDR_DoAlarm	$CFD7	Die Weckroutine
SetADDR_GetFiles	$CFD4	Dateiauswahlbox
SetADDR_GFilData	$CFD1	GetFiles: Einlesen der Tabelleneinträge
SetADDR_GFilMenu	$CFCE	GetFiles: Zeichnen Auswahlbox-Menüs
SetADDR_DB_SCRN	$CFCB	Wird eine Dialogbox geöffnet so rettet MegaPatch den Inhalt des Bildschirms in einen geschützten Bereich in der Speicher-erweiterung. Nach beenden der Dialogbox kopiert diese Routine den Bildschirminhalt wieder in den Vordergrundbildschirm.
SetADDR_DB_GRFX	$CFC8	Grafikdaten unter Dialogbox.
SetADDR_DB_COLS	$CFC5	Farbdaten unter Dialogbox.
SetADDR_BackScrn	$CFC2	Die Routine, welche das Hintergrundbild in den Vordergrundbildschirm kopiert. Programmierer müssen diese Routine nicht selbst ausführen, dafür existiert ein Einsprung in der Sprungtabelle.
SetADDR_ScrSaver	$CFBF	Der aktuelle Bildschirmschoner.

Tabelle 9: Sprungtabelle für ausgelagerte Kernal-Routinen (FORTSETZUNG)

Routine	Adr.	Beschreibung
SetADDR_Spooler	$CFBC	Spooler-Menü.
SetADDR_PrnSpool	$CFB9	Zeigt auf Druckertreiber für Spooler.
SetADDR_PrnSpHdr	$CFB6	Der Infoblock des Druckerspoolers. 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.
SetADDR_Printer	$CFB3	Der aktuelle Druckertreiber. Dieser muß sich nicht mehr auf jeder Diskette befinden sondern wird nun wie beim C128 ständig im Speicher gehalten.
SetADDR_PrntHdr	$CFB0	Der Infoblock zum aktuellen Druckertreiber.

HINWEIS: Ein Teil dieser Sprungbefehle ist für den Anwender nicht weiter von Bedeutung und sind nur der Vollständigkeit halber hier aufgeführt.

(7) LAUFWERKSTREIBER

Standard-Einsprungadressen

Diese Einsprungadressen sind unverändert und gelten ab sofort für alle Laufwerkstreiber (auch für Standard-1541-Laufwerke):

Tabelle 10: Einsprungtabelle Laufwerkstreiber

```
:InitForIO              w xInitForIO
:DoneWithIO             w xDoneWithIO
:ExitTurbo              w xExitTurbo
:PurgeTurbo             w xPurgeTurbo
:EnterTurbo             w xEnterTurbo
:ChangeDiskDev          w xChangeDiskDev
:NewDisk                w xNewDisk
:ReadBlock              w xReadBlock
:WriteBlock             w xWriteBlock
:VerWriteBlock          w xVerWriteBlock
:OpenDisk               w xOpenDisk
:GetBlock               w xGetBlock
:PutBlock               w xPutBlock
:GetDirHead             w xGetDirHead
:PutDirHead             w xPutDirHead
:GetFreeDirBlk          w xGetFreeDirBlk
:CalcBlksFree           w xCalcBlksFree
:FreeBlock              w xFreeBlock
:SetNextFree            w xSetNextFree
:FindBAMBit             w xFindBAMBit
:NxtBlkAlloc            w xNxtBlkAlloc
:BlkAlloc               w xBlkAlloc
:ChkDkGEOS              w xChkDkGEOS
:SetGEOSDisk            w xSetGEOSDisk

:Get1stDirEntry         jmp xGet1stDirEntry
:GetNxtDirEntry         jmp xGetNxtDirZntry
:GetBorderBlock         jmp xGetBorderBlock
:CreateNewDirBlk        jmp xCreateNewDirBlk
:GetBlock_dskBuf        jmp xGetBlock_dskBuf
:PutBlock_dskBuf        jmp xPutBlock_dskBuf
:TurboRoutine_r1        jmp xTurboRoutine_r1
:GetDiskError           jmp xGetDiskError
:AllocateBlock          jmp xAllocateBlock
:ReadLink               jmp xReadLink

;Kennbyte für Laufwerkstreiber
:DiskDrvType            b DiskDrvMode
:DiskDrvVersion         b DriverVersion
```

Die beiden Einsprünge :TurboRoutine_r1 und :GetDiskError werden von den internen TurboDOS-Routinen verwendet. Für Programmierer sind diese Routinen ohne Bedeutung.

Das Register :DiskDrvType wurde vom Gateway übernommen: Hier findet man den Laufwerkstyp wie er auch in :RealDrvType abgelegt ist. Im Register :DiskDrvVersion findet man die Versions-Nr. des aktuellen Treibers. Man könnte meinen das hier eine durchgehende Numerierung 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.

Erweiterte Einsprungadressen

Zuerst eine übersicht der neuen Routinen. Teilweise sind diese schon in den Laufwerkstreibern des GateWay-Systems enthalten:

Tabelle 12: Einsprungtabelle für NativeMode-Funktionen

```
:OpenRootDir            jmp xOpenRootDir        ;GateWay
:OpenSubDir             jmp xOpenSubDir         ;GateWay
:GetBAMBlock            jmp xGetBAMBlock        ;GateWay
:PutBAMBlock            jmp xPutBAMBlock        ;GateWay
```

Erweiterte Funktionen

```
:GetPDirEntry           jmp xGetPDirEntry
:ReadPDirEntry          jmp xReadPDirEntry
:OpenPartition          jmp xOpenPartition
:SwapPartition          jmp xSwapPartition
:GetPTypeData           jmp xGetPTypeData
:SendCommand            jmp xSendCommand
```

Kennung für erweiterte MegaPatch-Laufwerkstreiber

```
:DiskDrvTypeCode        b "MPDD3",NULL      ;Ab $906E
```

Diese Einsprünge sind in allen MegaPatch-Laufwerkstreibern enthalten. Um nun festzustellen, ob der aktuelle Treiber ein MegaPatch-Treiber ist, wurde die Kennung "MPDD3" integriert. Dieses Kürzel steht für "MegaPatchDiskDriver Version 3".

Ab MegaPatch V3.3r6 gibt es zusätzliche eingeführte Register und Einsprungadressen. Um feststellen zu können ob ein solcher Treiber vorliegt, wurde eine weitere ASCII-Kennung integriert:

```
:DDX                    b "DDX",NULL        ;Ab $9074
```

Ist diese Kennung ab $9074 vorhanden, dann unterstützt der Treiber die folgenden, teilweise bereits in früheren Versionen vorhandenen, neuen Adressen und Funktionen.

Adresse ":Flag_SD2IEC" $(9078, 1 Byte)

Diese Adresse wurde inn V3.3r4 eingefügt und in V3.3r6 verschoben, um innerhalb des Laufwerktreibers den Wert für ":RealDrvMode" zu setzen. Der Wert wird nur innerhalb des 1541/71/81 und SD2IEC-Treibers verwendet, ist aber aus Kompilitätsgründen in allen Treibern enthalten.
Das Flag wird durch die Installationsroutine des Laufwerkstreibers ermittelt und bei der Installation direkt im Laufwerkstreiber gespeichert. Da der Wert mit dem Assembler-Befehl "ORA" direkt mit ":RealDrvMode" verknüpft wird, ist ":Flag_SD2IEC" gleich %00000010 für "SD2IEC" oder %00000000 für "Kein SD2IEC". Der Wert wird in der Routine ":EnterTurbo" mit ":RealDrvMode" verknüpft.

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

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

Adresse ":GeoRAMBSize" $(9079, 1 Byte)

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

Wert	Bankgröße	Größe der GeoRAM mit 256 Speicherseiten
$10	16 KByte	Max. 4.096 KByte
$20	32 KByte	Max. 8.192 KByte
$40	64 KByte	Max. 16.384 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.

Adresse ":DDX_EXT_DATA1/2" $(907A, 2 Bytes)

Diese Adressen sind ab V3.3r6 in allen Laufwerkstreibern zu finden und sind für künftige Anwendungen reserviert. Die Adressen werden innerhalb von V3.3r6 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 dauerhaft im Treiber innerhalb des erweiterten GEOS-DACC zu speichern.

Hinweis:
Die beiden Adressen befinden sich innerhalb des Laufwerktreibers!
Wenn die Werte durch die Anwendung nicht permanent im Treiber/GEOS-DACC gespeichert werden, dann wird der Wert auf den Standard-Wert ($00 = Wert nicht initialisiert) zurückgesetzt, wenn der Treiber oder das Laufwerk gewechselt wird. Daher empfiehlt es sich den Wert $00 zu vermeiden.

Wenn der Wert permanent im Treiber des laufenden Systems 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 HDMP) wird diese beiden Adressen immer auf den Wert $00 zurücksetzen. Ansonsten wird GEOS/MegaPatch 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.

GeoDesk64 ist eine erste Anwendung, die auf diese Adressen aufsetzt:
Die Werte werden dazu genutzt um zwischenzeitliche Laufwerkswechsel zu erkennen. Wenn das Laufwerk nicht gewechselt wurde, dann können bestimmte Funktionen übersprungen werden, was zu einer geringen Verbesserung der Reaktionszeit innerhalb des Programms führt.

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.

Ein möglicher Workaround wäre, die Adressen vor dem starten einer externen Anwendung zwischenzuspeichern, die Anwendung zu starten, und nach der Rückkehr die Werte wieder zurückzusetzen.

Adresse ":DDrvNMData" $(9082, 8 Bytes)

Interne Datenregister der NativeMode-Laufwerkstreiber. Die hier abgelegten Daten werden der Vollständigkeit halber im folgenden beschrieben.
Bis zur Version MegaPatch V3.3r6 waren die Adressen innerhalb der NativeMode-Laufwerkstreiber nicht immer identisch, da diese nicht standardisiert sind. Ab V3.3r6 folgenden diese Werte im Anschluß an die erweiterten DDX-Funktionen, was sich aber künftig wieder ändern kann.

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

Adresse ":DiskSize_Lb" $(9082, 1 Byte)
Adresse ":DiskSize_Hb" $(9083, 1 Byte)

Hier legt die Routine :OpenDisk die Größe der aktuellen Partition bzw. RAM-Laufwerk in Kb im LOW/HIGH-Format ab. Der Wert wird von :CalcBlksFree verwendet um den noch freien Speicher auf dem Laufwerk zu ermitteln. Dazu wird dieser Wert / 4 geteilt und in :r3 abgelegt.
Damit zählt :CalcBlksFree bei NativeMode-Laufwerken nicht wirklich die Anzahl der noch freien Blöcke, sondern gibt aus Gründen der höheren Geschwindigkeit nur einen ungefähren Wert zurück.
HINWEIS: Interne Adresse, kann sich in künftigen Versionen ändern.

Adresse ":LastTrOnDsk" $(9084, 1 Byte)

Hier wird durch :OpenDisk die letzte verfügbare Spur auf der aktuellen 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 NativeMode-Laufwerkstreiber diesen Wert hier ab um nicht jedes mal den zweiten BAM-Sektor einlesen zu müssen.
Das bedeutet auch das bei einem Partitionswechsel mit unterschiedlicher Partitionsgröße zwingend :OpenDisk aufzurufen ist um diesen Wert zu aktualisieren.
HINWEIS: Interne Adresse, kann sich in künftigen Versionen ändern.

Adresse ":DirHead_Tr" $(9085, 1 Byte)
Adresse ":DirHead_Se" $(9086, 1 Byte)

Hier wird von der Routine Adresse :OpenRootDir bzw. :OpenSubDir der erste BAM-Sektor $01/$01 (ROOT) bzw. 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 Sektors geprüft ob der Disketten-Name an Byte $90 eingeblendet werden soll oder nicht. Dies erfolgt bei NativeMode und beim 1581-Format automatisch um kompatibel zum Format 1541/1571 zu bleiben.
HINWEIS: Interne Adresse, kann sich in künftigen Versionen ändern.

Adresse ":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.

Adresse ":CurSek_BAM" $(9088, 1 Byte)

Hier wird die Sektor-Adresse des BAM-Sektors in :dir3Head abgespeichert. :dir3Head 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.

Adresse ":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, falls dieser zuvor verändert wurde.
HINWEIS: Interne Adresse, kann sich in künftigen Versionen ändern.

Beschreibung der erweiterten Laufwerks-Routinen

Adresse ":OpenRootDir" ($9050)

Parameter:	keine	
Rückgabe:	***	***
Verändert:	AKKU, X, Y Register :r0 bis :r5	

Dieser Einsprung existiert nur in NativeMode-Treibern. Andere Treiber beenden diese Routine mit einem Fehler "ILLEGAL_DEVICE" ($40).
:OpenRootDir öffnet auf einem NativeMode-Laufwerke das Hauptverzeichnis. Es werden keine besonderen Parameter benötigt.

Adresse ":OpenSubDir" ($9053)

Parameter:	:r1L :r1H	Zeiger Verzeichnis/Track Zeiger Verzeichnis/Sektor
Rückgabe:	***	***
Verändert:	AKKU, X, Y Register :r0 bis :r5	

Öffnet auf NativeMode-Laufwerken ein Unterverzeichnis. Dazu muß man in :r1 den Track und Sektor des ersten Verzeichnis-Blocks übergeben. Diese Angaben kann man direkt aus dem Verzeichnis-Eintrag entnehmen.

Adresse ":GetBAMBlock" ($9056)

Parameter:	AKKU	BAM-Sektor Wert von #2 bis #33
Rückgabe:	***	***
Verändert:	AKKU, X, Y	

Ließt einen NativeBAM-Sektor in den Speicher. Hat für den Programmierer keinerlei Bedeutung, da diese Routine von den eigentlichen BAM-Routinen verwendet wird.

Adresse ":PutBAMBlock" ($9059)

Parameter:	AKKU	BAM-Sektor Wert von #2 bis #33
Rückgabe:	***	***
Verändert:	AKKU, X, Y	

Ähnlich :GetBAMBlock, aber der NativeBAM-Sektor wird auf Diskette aktualisiert.

Adresse ":GetPDirEntry" ($905C)

Parameter:	:r3H	Partitions-Nr.
	:r4	Ablagebereich (31 Bytes)
Rückgabe:	31 Bytes	
Verändert:	AKKU, X, Y	

Ließt einen Eintrag aus dem Partitions-Verzeichnis ein. Dazu übergibt man im Register :r3H die Partitions-Nr. und in :r4 einen Zeiger auf einen Speicherbereich in dem GetPDirEntry seine Daten (31 Bytes) ablegen kann.
Wichtig ist, das diese Routine vorher :ExitTurbo und :InitForIO aufruft. Zum Schluß wird noch :DoneWithIO aufgerufen.

Adresse ":ReadPDirEntry" ($905F)

Parameter:	:r3H	Partitions-Nr.
	:r4	Ablagebereich (31 Bytes)
Rückgabe:	31 Bytes	
Verändert:	AKKU, X, Y	

Entspricht :GetPDirEntry, jedoch muß der Anwender vorher die Routinen :ExitTurbo und :InitForIO manuell aufrufen. Diese Routine eignet sich nur für den Fall, das man mehr als einen Eintrag aus dem Partitions-Verzeichnis einlesen möchte. Zum Schluß muß man noch :DoneWithIO aufrufen.

Adresse ":OpenPartition" ($9062)

Parameter:	:r3H	Partitions-Nr.
Rückgabe:	***	***
Verändert:	AKKU, X, Y Register :r0 bis :r5	

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

Adresse ":SwapPartition" ($9065)

Parameter:	:r3H	Partitions-Nr.
Rückgabe:	***	***
Verändert:	AKKU, X, Y Register :r0 bis :r5	

Entspricht :OpenPartition, zuvor müssen jedoch die Routinen :ExitTurbo und :InitForIO aufgerufen werden. Zum Schluß muß man noch :DoneWithIO aufrufen. Diese Routine kann man in Zusammenhang mit :ReadPDirEntry verwenden.

Adresse ":GetPTypeData" ($9068)

Parameter:	:r4	Ablagebereich (256 Bytes)
Rückgabe:	256 Bytes	
Verändert:	AKKU, X, Y	

Diese Routine erstellt eine Tabelle mit gültigen Partitionstypen. 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 Speicherbereich ablegen. Dort werden dann für die Partitionen 0-255 die Partitionstyp-Kennbytes abgelegt.
Da außer der CMD-HD kein anderes Laufwerk bis zu 255 Partitionen unterstützt, werden die restlichen Bytes mit Null-Bytes aufgefüllt. Zu beachten ist, das Partitionstyp-Kennbytes im GEOS-Format und nicht im CMD-Format abgelegt werden.

Hier eine übersicht:

Partitionsformat	Kennbyte
Nicht erstellt	$00
1541-Emulations-Modus	$01
1571-Emulations-Modus	$02
1581-Emulations-Modus	$03
Native-Modus	$04
Direct Access Memory (DACC)	$07
System	$FF

Warum CMD bei den neuen CMD-Geräten die Reihenfolge geändert hat (dort entspricht NativeMode = Typ $01) ist unverständlich. :GetPTypeData wandelt diese Kennbytes in das GEOS-Format, so das in Verbindung mit dem Emulationsregister :driveType die passenden Partitionen ermittelt werden können.

Adresse ":SendFloppyCom" ($906B)

Parameter:	:r0	Zeiger auf Floppy-Befehl
	:r2L	Länge des Befehls in Bytes
Rückgabe:	***	***
Verändert:	AKKU, X, Y	

Diese Routine sendet einen Floppy-Befehl an das entsprechende Laufwerk. Auf RAM-Laufwerken erhält man hier einen "ILLEGAL_DEVICE"-Fehler ($40).
Vor dem Aufruf muß man in :r0 einen Zeiger auf den Floppy-Befehl ablegen. Da man das Ende nicht mit einem Null-Byte markieren kann, muß die Länge des Befehls im Register :r2L abgelegt werden.
Vorher muß man jedoch noch die Routinen :ExitTurbo und :InitForIO aufrufen. Am Ende ist :DoneWithIO aufzurufen.

Adresse ":InitForDDrvOp" ($907c)

Parameter:	***	***
Rückgabe:	r0	Adr. Laufwerkstreiber RAM
	r1	Adr. Laufwerkstreiber DACC
	r2	Größe Laufwerkstreiber
	r3L	Speicherbank in DACC
Verändert:	AKKU, X, Y	
	Register :r0 bis :r3L	

Diese Routine setzt die Speicheradressen für den aktuellen Laufwerkstreiber im C64- und GEOS-DACC-Speicher. Danach kann FetchRAM oder StashRAM verwendet werden, um den Treiber auszulesen oder die erweiterten DDX-Werte ab $9078 im GEOS-DACC zu speichern.

Adresse ":DoneWithDDrvOp" ($907f)

Parameter:	***	***
Rückgabe:	***	***
Verändert:	AKKU, X, Y	
	Register :r0 bis :r3L	

Nach ":InitForDDrvOp" muss immer ":DoneWithDDrvOp" aufgerufen werden, da die Register :r0L 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.

Eine Routine zum permanenten speichern der Werte in den neuen DDX-Adressen könnte wie folgt aussehen:

```
* CODE *
:UpdateGEOS  jsr InitForDDrvOp
             jsr StashRAM
             jsr DoneWithDDrvOp
```

HINWEIS: Die neuen DDX-Werte ab $9078 werden nur zum Teil automatisch im Treiber gespeichert, z.B. ":Flag_SD2IEC" oder ":GeoRAMBSize".
Die reservierten Adressen ":DDX_EXT_DATA1/2" können zwar von Anwendungsprogrammen genutzt werden, der GEOS-Kernal aktualisiert die Werte aber nicht automatisch im GEOS-DACC-Speicher.
Anwendungsprogramme müssen daher die obige Routine ":UpdateGEOS" in das Programm integrieren und nach dem ändern der DDX-Werte aufrufen.