

ExtraV34

COLLABORATORS

	<i>TITLE :</i> ExtraV34		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 31, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	ExtraV34	1
1.1	TABLE OF CONTENTS - DATABASE: ExtraV34.guide	1
1.2	*** Legal-ReadMe ***	1
1.3	*** Haftung ***	4
1.4	*** Hinweise ***	4
1.5	*** Details ***	5
1.6	*** Danksagung ***	5
1.7	*** extra.library / BAMChecksum() (OFFSET -42) V1.323 ***	6
1.8	*** extra.library / BootBlockSum() (OFFSET -48) V1.323 ***	6
1.9	*** extra.library / DiskBlockSum() (OFFSET -54) V1.323 ***	7
1.10	*** extra.library / OpenBuf() (OFFSET -60) V1.323 ***	7
1.11	*** extra.library / v2_OpenBuf() (OFFSET -300) V2.062 ***	8
1.12	*** extra.library / CloseBuf() (OFFSET -66) V1.323 ***	8
1.13	*** extra.library / FileToBuf() (OFFSET -72) V1.323 ***	9
1.14	*** extra.library / v2_FileToBuf() (OFFSET -306) V2.062 ***	10
1.15	*** extra.library / BufToFile() (OFFSET -78) V1.323 ***	10
1.16	*** extra.library / ClearBuf() (OFFSET -84) V1.323 ***	11
1.17	*** extra.library / DuplicateBuf() (OFFSET -90) V1.323 ***	11
1.18	*** extra.library / v2_DuplicateBuf() (OFFSET -312) V2.062 ***	12
1.19	*** extra.library / BufInfo() (OFFSET -96) V1.323 ***	12
1.20	*** extra.library / SeekBuf() (OFFSET -102) V1.323 ***	13
1.21	*** extra.library / Store() (OFFSET -108) V1.323 ***	14
1.22	*** extra.library / InsertBuf() (OFFSET -114) V1.323 ***	14
1.23	*** extra.library / InsertFile() (OFFSET -120) V1.323 ***	15
1.24	*** extra.library / Replace() (OFFSET -126) V1.323 ***	15
1.25	*** extra.library / Get() (OFFSET -132) V1.323 ***	16
1.26	*** extra.library / Clear() (OFFSET -138) V1.323 ***	16
1.27	*** extra.library / FindData() (OFFSET -144) V1.323 ***	17
1.28	*** extra.library / FindNext() (OFFSET -150) V1.323 ***	18
1.29	*** extra.library / FindPrev() (OFFSET -156) V1.323 ***	18

1.30	*** extra.library / FreeFinds() (OFFSET -162) V1.323 ***	18
1.31	*** extra.library / SetBookmark() (OFFSET -168) V1.323 ***	19
1.32	*** extra.library / GotoBookmark() (OFFSET -174) V1.323 ***	19
1.33	*** extra.library / FindString() (OFFSET -180) V1.323 ***	20
1.34	*** extra.library / GetLength() (OFFSET -186) V1.323 ***	21
1.35	*** extra.library / GetLastError() (OFFSET -192) V1.323 ***	21
1.36	*** extra.library / LongToString() (OFFSET -198) V1.323 ***	22
1.37	*** extra.library / StringToLong() (OFFSET -204) V1.323 ***	22
1.38	*** extra.library / PackByteRun1() (OFFSET -210) V1.323 ***	23
1.39	*** extra.library / UnpackByteRun1() (OFFSET -216) V1.323 ***	23
1.40	*** extra.library / PrintIoError() (OFFSET -222) V1.323 ***	24
1.41	*** extra.library / FlipCase() (OFFSET -228) V1.323 ***	24
1.42	*** extra.library / CmpMem() (OFFSET -234) V1.323 ***	25
1.43	*** extra.library / CopyMemI() (OFFSET -240) V1.323 ***	25
1.44	*** extra.library / FillMem() (OFFSET -246) V1.323 ***	26
1.45	*** extra.library / GetMem() (OFFSET -252) V1.323 ***	26
1.46	*** extra.library / UnGetMem() (OFFSET -258) V1.323 ***	27
1.47	*** extra.library / OutlineOff() (OFFSET -264) V1.323 ***	27
1.48	*** extra.library / OutlineOn() (OFFSET -270) V1.323 ***	27
1.49	*** extra.library / SetOPen() (OFFSET -276) V1.323 ***	27
1.50	*** extra.library / SpecText() (OFFSET -282) V1.323 ***	28
1.51	*** extra.library / SpecTextPrefs() (OFFSET -288) V1.323 ***	29
1.52	*** extra.library / ShowMsg() (OFFSET -294) V1.323 ***	30
1.53	*** extra.library / SMCOLOR() ***	32
1.54	*** extra.library / SMTEXT() ***	32
1.55	*** extra.library / SMTEXT() ***	33
1.56	*** Ivo / Lib-Call-Macro ***	34

Chapter 1

ExtraV34

1.1 TABLE OF CONTENTS - DATABASE: ExtraV34.guide

Willkommen zur AmigaGuide® - Dokumentation der »extra.library« !!

 Vorbemerkungen

1.) Legal-ReadMe 2.) Haftung 3.) Hinweise 4.) Details 5.) Danksagung

Die Funktionen ab V1.323

BAMCheckSum() BootBlockSum() DiskBlockSum()
 OpenBuf() CloseBuf() FileToBuf() BufToFile()
 ClearBuf() DuplicateBuf() BufInfo() SeekBuf()
 Store() InsertBuf() InsertFile() Replace()
 Get() Clear() FindData() FindNext()
 FindPrev() FreeFinds() SetBookmark() GotoBookmark()
 FindString() GetLength() LastError() LongToString()
 StringToLong() PackByteRun1() UnpackByteRun1() PrintIoError()
 FlipCase() CmpMem() CopyMemI() FillMem()
 GetMem() UnGetMem() OutlineOff() OutlineOn()
 SetOPen() SpecText() SpecTextPrefs() ShowMsg()

Die Funktionen ab V2.062

v2_OpenBuf() v2_FileToBuf() v2_DuplicateBuf()

Die Macros

SMCOLOR() SMTEXTP() SMTEXT() Lib-Call-Macro

1.2 *** Legal-ReadMe ***

```

;=====
;
; |*****|
; |*|-----|*|-----|*|
; |*|          |*| $Id: extra.library (28.04.01)          |*|

```

```

; |*| ##### ##### |*| RhoSigma Source based on NDK 3.1 Includes 40.15 |*|
; |*| ##  ##  ##  # |*| _____ |*|
; |*| ##  ##  ##  |*| |*| |*|
; |*| #####  ##  |*| Shared library with OS supporting functions |*|
; |*| ##      ##  |*| |*| |*|
; |*| ##      ##  # |*| ----- |*|
; |*| ##      ##### |*| Copyright © 1998-2001 RhoSigma, Roland Heyder |*|
; |*| |*| |*| All Rights Reserved. |*|
; |*| _____ |*| _____ |*|
; |*****|
;
; Die folgenden Worte beziehen sich auf das Projekt »extra.library«. Dieses
; Projekt beinhaltet folgende Dateien:
;
; Verzeichnis: Extra
; -----
;  ReadMe          - was Sie gerade lesen
;  Install         - kopiert die Library ins LIBS:-Verzeichnis
;
; Verzeichnis: Extra/libs
; -----
;  extra.lib_ocs   - der Programmcode (eine Shared-Library) OCS/ECS/AGA
;  extra.lib_cgxx  - die CyberGraphX-fähige Variante der Library
;
; Verzeichnis: Extra/include/rhosigma
; -----
;  extra_lib.fds   - Funktions- und Parameterdefinitionen
;
;  extradefs.i     - Definition von Konstanten (Assembler)
;  extra_lib.i     - die Library-Vector-Offsets (Assembler)
;
;  extradefs.h     - Definition von Konstanten (ANSI-C/C++)
;  extra_lib.h     - die amical/libcall-Pragmas (ANSI-C/C++)
;  extra_protos.h  - die Prototypen der Funktionen (ANSI-C/C++)
;
;  extra.bh        - Header-File für div. BASICS (HBC-Compiler)
;  extra.bc        - Constants-File für div. BASICS (HBC-Compiler)
;  extra.bmap      - Hilfsdatei für div. BASICS
;
;  settings.i      - weitere globale Definitionen (Assembler)
;  settings.h      - weitere globale Definitionen (ANSI-C/C++)
;  settings.bc     - weitere globale Definitionen für BASIC
;
;  rhosigma.c      - Source-Datei mit zusätzlichen Hilfs-Funktionen
;  rhosigma.h      - Prototypen & Defines für rhosigma.c (ANSI-C/C++)
;
; Verzeichnis: Extra/Docs
; -----
;  Extra.DOC       - Dokumentation extra.library (ASCII-Datei)
;  ExtraV34.guide  - Dokumentation extra.library (AmigaGuide® für OS 1.3+)
;  ExtraV39.guide  - Dokumentation extra.library (AmigaGuide® für OS 3.0+)
;
; Verzeichnis: Extra/Demos
; -----
;  ShowMsgDemo.ASM - Beispiel für »extradefs.i« Assembler-Macros
;  ShowMsgDemo.C   - Beispiel für »extradefs.h« Macros & Funktionen (ANSI-C/C++)
;  ShowMsgDemo.BAS - Beispiel für »extra.bc« BASIC-Funktionen

```



```

;           E-MAIL: rhosigma@crosswinds.net
;
; P.S. - Disk und Versandkosten sind bereits in der Share-Gebür enthalten !!
;       Wenn Sie die Vollversion via E-mail erhalten wollen, dann stellen
;       Sie bitte unbedingt sicher, daß Sie Ihre E-mail-Adresse richtig an-
;       geben (Groß-/Kleinschreibung, Leerstellen, Punkte etc.). Bei E-mail-
;       Versandt reduziert sich außerdem die Share-Gebür auf DM 10,- da die
;       Disk- u. Portokosten entfallen.
;
; Nun viel Spaß mit der »extra.library« und vielen Dank für Ihr Interesse.
;=====

```

1.3 *** Haftung ***

```

;=====
; Hiermit lehne ich ausdrücklich jegliche Haftung bzw. Verantwortung für
; Konsequenzen aller Art, die aus dem Gebrauch des Projektes »extra.library«
; resultieren, strikt ab.
;
; Die »extra.library« wird Ihnen von mir so zur Verfügung gestellt, wie
; Sie sie vorgefunden haben. Das heißt, ich garantiere nicht, daß eventuelle
; Programmierfehler behoben werden, daß neue Versionen erscheinen, oder daß
; Sie Hilfe erhalten, falls bei Ihnen ein Fehler auftritt.
;
; Obwohl die »extra.library« gründlich auf »Herz und Nieren« getestet wur-
; de, kann ich die Möglichkeit nicht ausschließen, daß die Library:
; - aus irgendwelchen Gründen nicht kompatibel zu Ihrem Rechner ist
; - Fehler enthält, die nur auf Ihrem Rechner auftreten
; - auf Ihrem Rechner nicht die erwarteten Resultate liefert
;
; Es liegt ausschließlich in Ihrer Verantwortung, alle notwendigen Sicher-
; heitsmaßnahmen zu treffen, um sich vor jeglichen schädlichen Auswirkungen
; bei der Nutzung der »extra.library« zu schützen.
;=====

```

1.4 *** Hinweise ***

```

;=====
; - Sollten Sie im folgenden Text auf die Zeichenkombination »(+0)« treffen,
; dann ist damit gemeint, daß der angegebene Parameter mit einem 0-Byte
; abgeschlossen werden muß (z.B. Dateinamen etc.).
; Wenn es sich jedoch um das Ergebnis einer Funktion handelt, dann können
; Sie bei dieser Angabe davon ausgehen, daß dieses 0-terminiert ist.
;
; - Parameter des Typs BOOL sind normaler Weise TRUE bzw. FALSE. Ich habe
; jedoch noch 3 neue Werte »erfunden« (UPPER/EQUAL/LOWER), deren Definition
; Sie im mitgelieferten Includefile »settings.i« finden.
;
; - Für alle Parameter, die mit »An!« bzw. »Dn!« angegeben sind, tragen Sie
; die volle Verantwortung, daß es sich wirklich um die von der Funktion
; benötigten Daten handelt. Alle anderen Parameter werden intern überprüft.
;
;

```

```

; - Bei Funktionen, bei denen das Ergebnis mit »D0*« angegeben ist, handelt
;   es sich um Ergebnistestende Funktionen (s.a. Abschnitt Details).
;
; ACHTUNG: Die normalen Regeln beim Umgang mit Shared-Libraries sollten
; ----- immer beachtet werden. Das heißt vor allem, daß Sie die Library
;           vor der Benutzung öffnen, und sich den Basiszeiger nicht etwa
;           aus der ExecBase.LibList holen. Einige Funktionen reagieren mit
;           dem DeadEnd-Alert »EL_Trespassed« (s. extradefs.i), wenn Sie den
;           Zugriff nicht vorher angemeldet haben.
;
; Zum Ende möchte ich noch sagen, daß Sie hier eine recht umfangreiche Doku-
; mentation vor sich haben, und wo viel Text ist, da sind sicher auch viele
; Fehler. Darum möchte ich Sie bitten, daß wenn Sie mich kontaktieren, dann
; nicht nur um mir zu sagen, daß ich hier oder dort einen Schreibfehler ge-
; macht habe !!
;=====

```

1.5 *** Details ***

```

;=====
; Hier noch einige technische Hintergrundinformationen für Interessierte.
;-----
; - Die Library ist kein Linkerprodukt aus Hochsprachen-Modulen, sondern
;   zu 100% in Assembler (DevPac 3.04) verfasst.
;
; - Keine Scratch-Register: bei allen Funktionen bleiben sämtliche Register-
;   inhalte, außer der des Rückgaberegisters D0, erhalten (es sei denn, daß
;   die Funktion kein Ergebnis liefert, dann bleibt D0 auch erhalten).
;
; - Alle Funktionen, bei denen beim Ergebnis »D0*« angegeben ist, testen das
;   Ergebnis unmittelbar vorm Return auf NULL (tst.l d0).
;   Da der Returnbefehl (rts) die Flags nicht verändert, können Assembler-
;   programmierer nach dem Funktionsaufruf sofort den Condition-Code auswerten,
;   wenn die Funktion zurückkehrt, und brauchen nicht noch selbst einen Test-
;   befehl zu programmieren.
;
; - Zum Teil hochoptimierter Code (auch auf 68000er CPU), vor allem bei zeit-
;   kritischen Funktionen.
;
; - Effektive Verwaltung der intern benötigten Speicherressourcen, wodurch
;   die meisten Funktionen auch bei stark fragmentierten Speicherlisten noch
;   immer lauffähig sind.
;
; - Das dynamische Puffersystem (DBS) sollte allen Programmierern von Editoren,
;   Datei-Managern o.ä. ein willkommenes Hilfsmittel sein. Der Speicher wird,
;   wie der Name schon sagt, dynamisch verwaltet, d.h. wie die RAM DISK, so
;   wird auch jeder Puffer in seiner Größe der gerade zu verwaltenden Daten-
;   menge (bis auf 4KB) genau angepasst.
;=====

```

1.6 *** Danksagung ***

```

;=====
; Vielen Dank schon jetzt an alle die, die sich trotz der allgemein schlech-
; ten Zahlungsmoral bei derartigen Produkten registrieren lassen werden. Sie
; motivieren mich dadurch, die Library weiter zu pflegen und zu erweitern.
;=====

```

1.7 *** extra.library / BAMCheckSum() (OFFSET -42) V1.323 ***

```

;=====
; Diese Funktion berechnet die Checksumme eines Bitmap-Blocks innerhalb
; eines der gängigen Amiga-FileSystems (DOS/0 - DOS/5), welches eine Block-
; gröÙe von 512 Bytes verwendet (z.B. normale Disketten).
;-----
; Synopsis:      success = BAMCheckSum (Block)
;                D0*                A0!
;
; Eingaben:     A0 --> APTR auf die Startadresse des Blocks (gerade Adresse,
;                sonst folgt Fehler)
;
; Ergebnis:     D0 --> BOOL - TRUE , wenn alles in Ordnung
;                - FALSE, wenn Startadresse ungerade war
;
; Bemerkung:    Die Summe wird gleich ordnungsgemäß in den Block geschrieben,
;                so daß er anschließend sofort gespeichert werden kann.
;
; Siehe auch:  BootBlockSum(), DiskBlockSum()
;=====

```

1.8 *** extra.library / BootBlockSum() (OFFSET -48) V1.323 ***

```

;=====
; Diese Funktion berechnet die Checksumme eines Boot-Blocks innerhalb eines
; der gängigen Amiga-FileSystems (DOS/0 - DOS/5), welches eine Blockgröße
; von 512 Bytes verwendet (z.B. normale Disketten).
;-----
; Synopsis:      success = BootBlockSum (Block)
;                D0*                A0!
;
; Eingaben:     A0 --> APTR auf die Startadresse des Blocks (gerade Adresse,
;                sonst folgt Fehler)
;
; Ergebnis:     D0 --> BOOL - TRUE , wenn alles in Ordnung
;                - FALSE, wenn Startadresse ungerade war
;
; Bemerkung:    Beachten Sie, daß den Boot-Block eigentlich zwei hinterein-
;                ander auf der Diskette liegende Blöcke darstellen (Block 0-1).
;                Auch bei der Berechnung müssen beide Blöcke unmittelbar hin-
;                tereinander im Speicher liegen.
;                Die Summe wird gleich ordnungsgemäß in den Block geschrie-
;                ben, so daß er anschließend sofort gespeichert werden kann.
;
;
;=====

```

```
; Siehe auch: BAMCheckSum(), DiskBlockSum()
```

```
;=====
```

1.9 *** extra.library / DiskBlockSum() (OFFSET -54) V1.323 ***

```
;=====
; Diese Funktion berechnet die Checksumme eines beliebigen Blocks innerhalb
; eines der gängigen Amiga-FileSystems (DOS/0 - DOS/5), welches eine Block-
; grÖÙe von 512 Bytes verwendet (z.B. normale Disketten), der KEINEN Boot-Block
; und KEINEN Bitmap-Block darstellt.
```

```
;-----
```

```
; Synopsis:      success = DiskBlockSum (Block)
;                D0*                A0!
```

```
; Eingaben:      A0 --> APTR auf die Startadresse des Blocks (gerade Adresse,
;                sonst folgt Fehler)
```

```
; Ergebnis:      D0 --> BOOL - TRUE , wenn alles in Ordnung
;                - FALSE, wenn Startadresse ungerade war
```

```
; Bemerkung:     Die Summe wird gleich ordnungsgemäß in den Block geschrieben,
;                so daß er anschließend sofort gespeichert werden kann.
```

```
; ACHTUNG:       Handelt es sich um ein FastFile-System (DOS/1, DOS/3, DOS/5),
; -----       dann darf diese Funktion NICHT auf Data-Blöcke angewendet
;                werden, da diese beim FFS keine Checksummen enthalten.
;                Nichtbeachtung führt zu Datenverlust, für den ich KEINERLEI
;                HAFTUNG übernehme !!
```

```
; Siehe auch:   BAMCheckSum(), BootBlockSum()
```

```
;=====
```

1.10 *** extra.library / OpenBuf() (OFFSET -60) V1.323 ***

```
;=====
; Diese Funktion baut einen neuen dynamischen Puffer auf, und führt alle
; nötigen Initialisierungen durch. Als Ergebnis erhalten Sie einen Zeiger,
; der bei allen Pufferfunktionen als Identifikation dieses Puffers dient.
```

```
;-----
```

```
; Synopsis:      bufHandle = OpenBuf (VOID)
;                D0*
```

```
; Ergebnis:      D0 --> APTR - auf den initialisierten BufHandle
;                - 0-PTR, wenn Fehler (Info mit LastError())
```

```
; Bemerkung:     Jeder geöffnete Puffer sollte, wenn er nicht mehr benötigt
;                wird, mittels CloseBuf() wieder geschlossen werden bzw.
;                wenigstens mittels der Funktion ClearBuf() auf minimalen
;                Speicherverbrauch reduziert werden.
```

```
; ACHTUNG:       Wenn Sie sich Puffer mit anderen Prozessen teilen, dann liegt
; -----       es ausschließlich in Ihrer Verantwortung, Zugriffe auf diese
```

```

;           Puffer so zu koordinieren und zu sichern, daß die Prozesse im
;           Rahmen des Multitaskings nicht gleichzeitig auf die Puffer zu-
;           greifen können !! - Achten Sie immer darauf, daß jede Puffer-
;           funktion vollständig abgeschlossen wurde, bevor ein anderer
;           Prozess die nächste startet.
;
; Siehe auch: CloseBuf(), FileToBuf(), BufToFile(), ClearBuf(),
;           DuplicateBuf(), BufInfo()
;
;-----
;--- Veränderungen seit v2.062 -----
;-----
;
; Bemerkung: Diese Funktion ist veraltet, bitte benutzen Sie stattdessen
;           die Funktion v2_OpenBuf(), welche zusätzlich die Angabe von
;           Speicher-Flags ermöglicht, um z.B. Audio- oder Grafik-Daten
;           gezielt ins CHIP-Ram zu verlagern.
;
; ACHTUNG: Diese Funktion hat und wird immer nur PUBLIC-Memory für den
;           Datenbereich verwenden, und unterliegt der durch das System
;           festgelegten Zuordnungsreihenfolge (meist FAST vor CHIP) !!
;
; Siehe auch: v2_FileToBuf(), v2_DuplicateBuf()
;=====

```

1.11 *** extra.library / v2_OpenBuf() (OFFSET -300) V2.062 ***

```

;=====
; Diese Funktion baut einen neuen dynamischen Puffer auf, und führt alle
; nötigen Initialisierungen durch. Dabei kann noch angegeben werden, welcher
; Speichertyp für den Datenbereich verwendet werden soll. Als Ergebnis erhalten
; Sie einen Zeiger, der bei allen weiteren Pufferfunktionen als Identifikation
; dieses Puffers dient.
;-----
; Synopsis:   bufHandle = v2_OpenBuf (MemType)
;            D0*                D0!
;
; Eingaben:   D0 --> ULONG eine Kombination von Speicher-Flags wie sie in
;            der Include-Datei »exec«/»memory.i« definiert sind.
;
; Ergebnis:   D0 --> APTR - auf den initialisierten BufHandle
;            - 0-PTR, wenn Fehler (Info mit LastError())
;
; Bemerkung:  Es gelten die gleichen Konditionen wie bei OpenBuf()
;
; Siehe auch: v2_FileToBuf(), v2_DuplicateBuf()
;=====

```

1.12 *** extra.library / CloseBuf() (OFFSET -66) V1.323 ***

```

;=====
; Diese Funktion schließt den angegebenen Puffer und gibt alle damit ver-

```

```

; bundenen Speicherressourcen an das System zurück. Nach Aufruf dieser Funk-
; tion können keine weiteren Pufferfunktionen mehr auf den angegebenen
; BufHandle ausgeführt werden.
;-----
; Synopsis:      success = CloseBuf (BufHandle)
;                D0*                A0
;
; Eingaben:      A0 --> APTR auf den zu schließenden BufHandle
;
; Ergebnis:      D0 --> BOOL - TRUE , wenn alles in Ordnung
;                - FALSE, wenn Fehler (Info mit LastError())
;
; ACHTUNG:      Wenn ein Task die »extra.library« ENDGÜLTIG schließt (es kann
; -----      ja sein, daß mehrere Open-Aufrufe verschachtelt wurden), dann
;                werden automatisch alle noch offenen Puffer, die DIESER TASK
;                geöffnet hat (egal, mit welcher Funktion) wieder geschlossen.
;                Diese Funktion löscht automatisch die Prozessor-Caches.
;
; Siehe auch:   OpenBuf(), FileToBuf(), BufToFile(), ClearBuf(),
;                DuplicateBuf(), BufInfo()
;=====

```

1.13 *** extra.library / FileToBuf() (OFFSET -72) V1.323 ***

```

;=====
; Diese Funktion lädt die Datei mit dem angegebenen Namen in einen neuen dy-
; namischen Puffer. Es wird der fertig initialisierte BufHandle des neu an-
; gelegten Puffers zurückgegeben. Die aktuelle Cursorposition steht nach dem
; Laden auf dem Pufferanfang (Pos. 0).
;-----
; Synopsis:      bufHandle = FileToBuf (FileName)
;                D0*                A0!
;
; Eingaben:      A0 --> STRPTR auf einen AmigaDOS-Dateinamen(+0)
;
; Ergebnis:      D0 --> APTR - auf den neu initialisierten BufHandle
;                - 0-PTR, wenn Fehler (Info mit LastError())
;
; Bemerkung:     Ein mittels dieser Funktion geöffneter Puffer wird ebenfalls
;                mit CloseBuf() wieder geschlossen.
;                Diese Funktion löscht automatisch die Prozessor-Caches.
;
; ACHTUNG:      Diese Funktion darf NUR von vollwertigen DOS-Prozessen auf-
; -----      gerufen werden. Ein normaler EXEC-Task ist nicht ausreichend.
;
; Siehe auch:   OpenBuf(), CloseBuf(), BufToFile(), ClearBuf(),
;                DuplicateBuf(), BufInfo()
;-----
; --- Veränderungen seit v2.062 -----
;-----
;
; Bemerkung:     Diese Funktion ist veraltet, bitte benutzen Sie stattdessen
;                die Funktion v2_FileToBuf(), welche zusätzlich die Angabe von
;                Speicher-Flags ermöglicht, um z.B. Audio- oder Grafik-Daten

```

```

;           gezielt ins CHIP-Ram zu laden.
;
;   ACHTUNG: Diese Funktion hat und wird immer nur PUBLIC-Memory für den
;   ----- Datenbereich verwenden, und unterliegt der durch das System
;           festgelegten Zuordnungsreihenfolge (meist FAST vor CHIP) !!
;
; Siehe auch: v2_OpenBuf(), v2_DuplicateBuf()
;=====

```

1.14 *** extra.library / v2_FileToBuf() (OFFSET -306) V2.062 ***

```

;=====
; Diese Funktion lädt die Datei mit dem angegebenen Namen in einen neuen dy-
; namischen Puffer. Dabei kann noch angegeben werden, welcher Speichertyp für
; den Datenbereich verwendet werden soll. Es wird der fertig initialisierte
; BufHandle des neu angelegten Puffers zurückgegeben. Die aktuelle Cursorpo-
; sition steht nach dem Laden auf dem Pufferanfang (Pos. 0).
;-----
; Synopsis:   bufHandle = v2_FileToBuf (FileName, MemType)
;             D0*                A0!      D0!
;
; Eingaben:   A0 --> STRPTR auf einen AmigaDOS-Dateinamen(+0)
;             D0 --> ULONG  eine Kombination von Speicher-Flags wie sie in
;             der Include-Datei »exec«/»memory.i« definiert sind.
;
; Ergebnis:   D0 --> APTR - auf den neu initialisierten BufHandle
;             - 0-PTR, wenn Fehler (Info mit LastError())
;
; Bemerkung:  Es gelten die gleichen Konditionen wie bei FileToBuf()
;
; Siehe auch: v2_OpenBuf(), v2_DuplicateBuf()
;=====

```

1.15 *** extra.library / BufToFile() (OFFSET -78) V1.323 ***

```

;=====
; Diese Funktion schreibt den gesamten Datenbereich des durch den BufHandle
; spezifizierten Puffers in eine Datei mit dem angegebenen Namen.
;-----
; Synopsis:   written = BufToFile (BufHandle, FileName)
;             D0*                A0      A1!
;
; Eingaben:   A0 --> APTR   auf den gewünschten BufHandle
;             A1 --> STRPTR auf einen AmigaDOS-Dateinamen(+0)
;
; Ergebnis:   D0 --> LONG - Anzahl der geschriebenen Bytes
;             - negativ, wenn Fehler (Info mit LastError())
;
; Bemerkung:  Ist beim Schreiben der Datei ein Fehler aufgetreten, dann wird
;             die unvollständige (fehlerhafte) Datei gleich wieder gelöscht.
;
;   ACHTUNG:  Diese Funktion darf NUR von vollwertigen DOS-Prozessen auf-

```

```
; ----- gerufen werden. Ein normaler EXEC-Task ist nicht ausreichend.
;
; Siehe auch: OpenBuf(), CloseBuf(), FileToBuf(), ClearBuf(),
;             DuplicateBuf(), BufInfo()
;=====
```

1.16 *** extra.library / ClearBuf() (OFFSET -84) V1.323 ***

```
;=====
; Diese Funktion löscht den gesamten Inhalt des angegebenen Puffers, und
; reduziert dessen Speicherbedarf auf ein Minimum von 4KB. Außerdem werden,
; wenn vorhanden, auch die internen Kopien aller Such-Daten wieder freige-
; geben sowie die gesetzten Bookmarks gelöscht. Der Puffer ist nach dieser
; Funktion also in dem Zustand, als wenn er gerade geöffnet worden wäre.
;-----
; Synopsis:      success = ClearBuf (BufHandle)
;                D0*                A0
;
; Eingaben:      A0 --> APTR auf den BufHandle des zu löschenden Puffers
;
; Ergebnis:      D0 --> BOOL - TRUE , wenn alles in Ordnung
;                - FALSE, wenn Fehler (Info mit LastError())
;
; Bemerkung:     Diese Funktion löscht automatisch die Prozessor-Caches.
;
; Siehe auch:   OpenBuf(), CloseBuf(), FileToBuf(), BufToFile(),
;               DuplicateBuf(), BufInfo()
;=====
```

1.17 *** extra.library / DuplicateBuf() (OFFSET -90) V1.323 ***

```
;=====
; Diese Funktion erstellt einen neuen dynamischen Puffer, dessen Daten eine
; genaue Kopie des angegebenen Quell-Puffers darstellen. Das heißt, es wer-
; den nicht nur die regulären Daten kopiert, sondern auch die Such-Daten so-
; wie die Bookmarks. Außerdem wird auch die aktuelle Cursorposition von dem
; Quell-Puffer übernommen. Sie erhalten als Ergebnis einen Zeiger auf den
; neu initialisierten BufHandle.
;-----
; Synopsis:      bufHandle = DuplicateBuf (BufHandle)
;                D0*                A0
;
; Eingaben:      A0 --> APTR BufHandle des zu duplizierenden Puffers
;
; Ergebnis:      D0 --> APTR - Zeiger auf den neu initialisierten BufHandle
;                - 0-PTR, wenn Fehler (Info mit LastError())
;
; Bemerkung:     Auch ein mittels dieser Funktion geöffneter Puffer wird mit
;                 der Funktion CloseBuf() wieder geschlossen.
;                 Diese Funktion löscht automatisch die Prozessor-Caches.
;
; Siehe auch:   OpenBuf(), CloseBuf(), FileToBuf(), BufToFile(),
;=====
```

```

;          ClearBuf(), BufInfo()
;
;-----
;--- Veränderungen seit v2.062 -----
;-----
;
; Bemerkung:   Diese Funktion ist veraltet, bitte benutzen Sie stattdessen
;              die Funktion v2_DuplicateBuf(), welche zusätzlich die Angabe
;              von Speicher-Flags ermöglicht, um z.B. Audio- oder Grafik-Daten
;              gezielt ins CHIP-Ram zu duplizieren.
;
; ACHTUNG:     Diese Funktion hat und wird immer DEN GLEICHEN Speichertyp für
;              den Datenbereich des neuen Puffers verwenden wie der originale
;              Puffer auch verwendet.
;
; Siehe auch:  v2_OpenBuf(), v2_FileToBuf()
;=====

```

1.18 *** extra.library / v2_DuplicateBuf() (OFFSET -312) V2.062 ***

```

;=====
; Diese Funktion erstellt einen neuen dynamischen Puffer, dessen Daten eine
; genaue Kopie des angegebenen Quell-Puffers darstellen. Das heißt, es wer-
; den nicht nur die regulären Daten kopiert, sondern auch die Such-Daten so-
; wie die Bookmarks. Außerdem wird auch die aktuelle Cursorposition von dem
; Quell-Puffer übernommen. Dabei kann noch angegeben werden, welcher Speicher-
; typ für den Datenbereich verwendet werden soll. Sie erhalten als Ergebnis
; einen Zeiger auf den neu initialisierten BufHandle.
;-----
; Synopsis:    bufHandle = v2_DuplicateBuf (BufHandle, MemType)
;              D0*                A0          D0!
;
; Eingaben:    A0 --> APTR  BufHandle des zu duplizierenden Puffers
;              D0 --> ULONG eine Kombination von Speicher-Flags wie sie in
;              der Include-Datei »exec«/»memory.i« definiert sind.
;
; Ergebnis:    D0 --> APTR - Zeiger auf den neu initialisierten BufHandle
;              - 0-PTR, wenn Fehler (Info mit LastError())
;
; Bemerkung:   Es gelten die gleichen Konditionen wie bei DuplicateBuf()
;
; Siehe auch:  v2_OpenBuf(), v2_FileToBuf()
;=====

```

1.19 *** extra.library / BufInfo() (OFFSET -96) V1.323 ***

```

;=====
; Mit dieser Funktion können Sie sich spezifische Daten über den momentanen
; Zustand eines Puffers holen. Als Ergebnis erhalten Sie einen Adress-Zeiger
; auf eine entsprechend ausgefüllte »BufInfoBlock«-Struktur (s. extradefs.i).
;-----
; Synopsis:    BIBStruct = BufInfo (BufHandle)

```

```

;                               D0*                               A0
;
; Eingaben:   A0 --> APTR auf den gewünschten BufHandle
;
; Ergebnis:   D0 --> APTR - auf ausgefüllte »BIBStruct« (s. extradefs.i)
;             - 0-PTR, wenn Fehler (Info mit LastError())
;
; Bemerkung:  Sie brauchen sich nicht um die Beschaffung bzw. Freigabe des
;             »BufInfoBlocks« zu kümmern, da der benötigte Speicherplatz
;             Ihrem Task beim Öffnen der »extra.library« zur Verfügung ge-
;             stellt wird, und dann beim Schließen der Library auch automa-
;             tisch wieder freigegeben wird. Beachten Sie jedoch, daß die
;             Struktur aufgrund dieser Verwaltungsmethode beim jeweils
;             nächsten Aufruf dieser Funktion wieder überschrieben wird.
;
; ACHTUNG:    In C und C++ ist das APTR-Ergebnis als Typ STRUCT BIBStruct*
;             zu behandeln.
;
; Siehe auch: OpenBuf(), CloseBuf(), FileToBuf(), BufToFile(),
;             ClearBuf(), DuplicateBuf()
;=====

```

1.20 *** extra.library / SeekBuf() (OFFSET -102) V1.323 ***

```

;=====
; Mit dieser Funktion können Sie die aktuelle Cursorposition innerhalb des
; angegebenen Puffers an jede beliebige Stelle setzen, und damit bestimmen,
; wo im Puffer die folgende(n) Pufferoperation(en) wirken soll(en). Es wird
; zuerst die durch den »Mode« angegebene Positionierung durchgeführt, und
; dann der Offsetwert »Position«, welcher sowohl positiv, als auch negativ
; sein darf, zu dieser Position hinzuaddiert.
;-----
; Synopsis:   oldPos = SeekBuf (BufHandle, Position, Mode)
;             D0*           A0           D0           D1
;
; Eingaben:   A0 --> APTR auf den gewünschten BufHandle
;             D0 --> LONG Offset, der zu der durch den »Mode« spezifizier-
;             ten Position addiert wird (positiv od. negativ)
;             D1 --> ULONG der zu verwendende »Mode« (s. extradefs.i)
;
; Ergebnis:   D0 --> LONG - alte Position, wenn alles in Ordnung
;             (ACHTUNG: Zählbeginn = 0 !!)
;             - negativ, wenn Fehler (Info mit LastError())
;
; Bemerkung:  Mit dieser Funktion können Sie außerdem schnell die aktuelle
;             Cursorposition ermitteln, wenn Ihnen der Weg über BufInfo()
;             zu aufwendig ist. Setzen Sie zu diesem Zweck den Parameter
;             »Mode« auf »SKM_Current«, und den Parameter »Position« auf
;             NULL. Mit diesem Aufruf bleibt die aktuelle Cursorposition
;             tatsächlich zwar unverändert, Sie bekommen sie aber auch als
;             angeblich alte Position zurückgeliefert.
;
; ACHTUNG:    Wenn ein Fehler auftritt, dann bleibt die alte Cursor-
;             position erhalten !!
;
;
;=====

```

```
; Siehe auch: Store(), InsertBuf(), InsertFile(), Replace(),
;             Get(), Clear()
;=====
```

1.21 *** extra.library / Store() (OFFSET -108) V1.323 ***

```
;=====
; Diese Funktion schreibt die angegebenen Daten in den durch den BufHandle
; spezifizierten Puffer. Die Daten werden an der aktuellen Cursorposition
; eingefügt. Eventuell schon vorhandene Daten an dieser Stelle werden ent-
; sprechend nach hinten verschoben. Die aktuelle Cursorposition wird auto-
; matisch nachgeführt, und steht nach dem Aufruf dementsprechend auf dem er-
; sten Byte nach den eingefügten Daten.
;-----
; Synopsis:      newPos = Store (BufHandle, Data, Size)
;                D0*                A0          A1!      D0
;
; Eingaben:      A0 --> APTR  auf den gewünschten BufHandle
;                A1 --> APTR  auf die einzufügenden Daten
;                D0 --> ULONG Menge der Daten in Bytes
;
; Ergebnis:      D0 --> LONG  - die neue Cursorposition, wenn alles in Ordnung
;                               (ACHTUNG: Zählbeginn = 0 !!)
;                               - negativ, wenn Fehler (Info mit LastError())
;
; Bemerkung:     Diese Funktion löscht automatisch die Prozessor-Caches.
;
; Siehe auch:   SeekBuf(), InsertBuf(), InsertFile(), Replace(),
;               Get(), Clear()
;=====
```

1.22 *** extra.library / InsertBuf() (OFFSET -114) V1.323 ***

```
;=====
; Diese Funktion fügt die Daten eines anderen, bereits bestehenden Puffers,
; an der aktuellen Cursorposition des gewünschten Puffers ein. Dies ist von
; Nutzen, wenn viele Einzeldaten am Anfang eines Puffers eingefügt werden
; sollen, was nämlich je nach Menge der nach hinten zu schiebenden Daten eine
; sehr zeitraubende Angelegenheit werden kann. In solchen Fällen öffnen Sie
; einfach einen neuen Puffer, wo Sie die Daten dann fortlaufend (also immer
; ans Pufferende anfügend) speichern können, wobei nichts nach hinten gescho-
; ben werden muß. Ist das getan, dann verwenden Sie diese Funktion, um die
; Daten nun in Ihren eigentlichen Arbeitspuffer einzufügen, womit jetzt nur
; einmal die ganzen Restdaten verschoben werden müssen. Die Cursorposition
; wird automatisch nachgeführt, und steht nach dem Aufruf dementsprechend auf
; dem ersten Byte nach den eingefügten Daten.
;-----
; Synopsis:      inserted = InsertBuf (BufHandle1, BufHandle2)
;                D0*                A0          A1
;
; Eingaben:      A0 --> APTR auf den gewünschten Ziel-BufHandle
;                A1 --> APTR auf den zu verwendenden Quell-BufHandle
;=====
```

```

;
; Ergebnis:      D0 --> LONG - Anzahl der eingefügten Bytes
;                - negativ, wenn Fehler (Info mit LastError())
;
; Bemerkung:     Diese Funktion löscht automatisch die Prozessor-Caches.
;
; Siehe auch:    SeekBuf(), Store(), InsertFile(), Replace(), Get(),
;                Clear()
;=====

```

1.23 *** extra.library / InsertFile() (OFFSET -120) V1.323 ***

```

;=====
; Diese Funktion fügt die Daten der angegebenen Datei in den gewünschten
; Puffer an der aktuellen Cursorposition ein. Die Cursorposition wird auto-
; matisch nachgeführt, und steht nach dem Aufruf dementsprechend auf dem er-
; sten Byte nach den eingefügten Daten.
;-----
; Synopsis:      inserted = InsertFile (BufHandle, FileName)
;                D0*                A0                A1!
;
; Eingaben:      A0 --> APTR   auf den gewünschten BufHandle
;                A1 --> STRPTR auf einen AmigaDOS-Dateinamen(+0)
;
; Ergebnis:      D0 --> LONG - Anzahl der eingefügten Bytes
;                - negativ, wenn Fehler (Info mit LastError())
;
; Bemerkung:     Diese Funktion löscht automatisch die Prozessor-Caches.
;
; ACHTUNG:      Diese Funktion darf NUR von vollwertigen DOS-Prozessen auf-
; -----      gerufen werden. Ein normaler EXEC-Task ist nicht ausreichend.
;
; Siehe auch:    SeekBuf(), Store(), InsertBuf(), Replace(), Get(),
;                Clear()
;=====

```

1.24 *** extra.library / Replace() (OFFSET -126) V1.323 ***

```

;=====
; Mit dieser Funktion können Sie die Daten an der aktuellen Cursorposition
; durch andere Daten ersetzen. Sollte die Länge der neuen Daten von der
; Länge der alten Daten abweichen, dann werden die nachfolgenden Daten im
; Puffer automatisch dementsprechend nach vorne oder hinten verschoben. Die
; aktuelle Cursorposition wird automatisch nachgeführt, und steht nach dem
; Aufruf dementsprechend auf dem ersten Byte hinter den neuen Daten.
;-----
; Synopsis:      newPos = Replace (BufHandle, Data, Size, OldSize)
;                D0*                A0                A1!    D0    D1
;
; Eingaben:      A0 --> APTR   auf den gewünschten BufHandle
;                A1 --> APTR   auf die neuen Daten
;                D0 --> ULONG  Menge der neuen Daten in Bytes
;

```

```

;           D1 --> ULONG Menge der alten Daten in Bytes
;
; Ergebnis:   D0 --> LONG - die neue Cursorposition, wenn alles in Ordnung
;              (ACHTUNG: Zählbeginn = 0 !!)
;              - negativ, wenn Fehler (Info mit LastError())
;
; Bemerkung:  Diese Funktion löscht automatisch die Prozessor-Caches.
;
; Siehe auch: SeekBuf(), Store(), InsertBuf(), InsertFile(),
;              Get(), Clear()
;=====

```

1.25 *** extra.library / Get() (OFFSET -132) V1.323 ***

```

;=====
; Mit dieser Funktion können Sie sich die physische Adresse der aktuellen
; Cursorposition besorgen. Diese Adresse können Sie dann für andere Funk-
; tionen verwenden (z.B. »Text()« der graphics.library, um die Daten auszu-
; geben). Die Daten brauchen also für solche Anwendungen nicht extra noch
; einmal ausgelesen werden (wär' ja auch unsinnig).
;-----
; Synopsis:   address = Get (BufHandle)
;              D0*           A0
;
; Eingaben:   A0 --> APTR auf den gewünschten BufHandle
;
; Ergebnis:   D0 --> APTR - auf die Daten an der aktuellen Cursorposition
;              - 0-PTR, wenn akt. Cursorposition = Pufferende
;              - negativ, wenn Fehler (Info mit LastError())
;
; Bemerkung:  Die gelieferte Adresse sollte NUR für lesende Zugriffe ver-
;              wendet werden, da alles was man »per Hand« an dieser Stelle
;              verändert nicht vom Puffersystem registriert wird.
;
; ACHTUNG:   Die ermittelte Adresse wird mit Aufruf der nächsten Puffer-
;             -----
;             funktion, welche Daten speichert od. löscht, wieder ungültig.
;             Eventuell worden die Daten dann nämlich verlagert.
;
; Siehe auch: SeekBuf(), Store(), InsertBuf(), InsertFile(),
;              Replace(), Clear()
;=====

```

1.26 *** extra.library / Clear() (OFFSET -138) V1.323 ***

```

;=====
; Diese Funktion löscht die angegebene Anzahl Datenbytes ab der aktuellen
; Cursorposition in dem durch den BufHandle spezifizierten Puffer. Eventuell
; noch dahinter stehende Daten werden wieder nach vorne aufgerückt. Die ak-
; tuelle Cursorposition bleibt unverändert, da die nachfolgenden Daten, auf
; denen die aktuelle Position nach der Operation ja steht, wieder nach vorne
; auf die Position der ehemaligen Daten aufrücken.
;-----

```

```

; Synopsis:      end = Clear (BufHandle, Size)
;               D0*          A0          D0
;
; Eingaben:      A0 --> APTR  auf den gewünschten BufHandle
;               D0 --> ULONG Anzahl der zu löschenden Datenbytes
;
; Ergebnis:      D0 --> BOOL - UPPER, wenn noch Daten aufgerückt sind
;               - EQUAL, wenn Löschkaktion Pufferende erreicht hat
;               - LOWER, wenn Fehler (Info mit LastError())
;
; Bemerkung:     Diese Funktion löscht automatisch die Prozessor-Caches.
;
; Siehe auch:    SeekBuf(), Store(), InsertBuf(), InsertFile(),
;               Replace(), Get()
;=====

```

1.27 *** extra.library / FindData() (OFFSET -144) V1.323 ***

```

;=====
; Mit dieser Funktion können Sie bestimmte Datenabschnitte innerhalb eines
; Puffers suchen. Die Suche beginnt dabei an der aktuellen Cursorposition.
; Werden die Daten gefunden, dann wird die gefundene Position automatisch
; zur aktuellen Cursorposition, im anderen Falle bleibt die alte Cursorposi-
; tion erhalten. Im Zusammenspiel mit den anderen Find-Funktionen und der
; Funktion Replace() kann man so z.B. einfach eine Search/Replace-Schleife
; realisieren. Der anzugebende Schlüsselwert dient bei späteren Aufrufen von
; FindNext()/FindPrev() als Identifikation der gewünschten Daten, welche
; vorher mit dieser Funktion initialisiert wurden. Es sind max. 8 Schlüssel-
; werte pro Puffer möglich. Sie könnten die ersten 4 Schlüssel z.B. verwen-
; den, um nach SPACES, TABS, LF's und FF's zu suchen, womit Sie auf einfache
; Art u. Weise eine Funktion hätten, mit der Sie den Cursor Wort-, Zeilen-
; oder auch Seitenweise weiter- bzw. zurückschalten könnten.
;-----
; Synopsis:      success = FindData (BufHandle, Key, Data, Size)
;               D0*          A0          D0   A1!   D1
;
; Eingaben:      A0 --> APTR  auf den gewünschten BufHandle
;               D0 --> ULONG Kennungs-Schlüssel dieser Daten (0-7 !!)
;               A1 --> APTR  auf die zu suchenden Daten
;               D1 --> ULONG Länge der zu suchenden Daten in Byte
;
; Ergebnis:      D0 --> BOOL - UPPER, wenn die Daten gefunden wurden
;               - EQUAL, wenn sie nicht gefunden wurden
;               - LOWER, wenn Fehler (Info mit LastError())
;
; Bemerkung:     Der Speicher Ihrer übergebenen Such-Daten kann nach Aufruf der
; Funktion anderweitig weiterverwendet werden, da die Such-Daten
; für die Verwendung mit den Funktionen FindNext()/FindPrev()
; intern kopiert und zwischengespeichert werden. Wenn schon Daten
; von einem früheren Aufruf dieser Funktion für den gleichen
; Schlüssel vorhanden sind, dann werden diese freigegeben und die
; jetzt angegebenen Daten an deren Stelle gespeichert.
; Diese Funktion löscht automatisch die Prozessor-Caches.
;
; Siehe auch:    FindNext(), FindPrev(), FreeFinds(), SetBookmark(),

```

```
; GotoBookmark()
;=====
```

1.28 *** extra.library / FindNext() (OFFSET -150) V1.323 ***

```
;=====
; Diese Funktion sucht das nächste Auftreten der Daten eines vorangegangenen
; FindData()-Aufrufs. Die Suche beginnt wie bei o.g. Funktion wieder an der
; aktuellen Cursorposition. Mit dem Schlüssel geben Sie an, von welchen Daten
; das nächste Auftreten gesucht werden soll.
;-----
; Synopsis: success = FindNext (BufHandle, Key)
;           D0*                A0      D0
;
; Eingaben: A0 --> APTR auf den gewünschten BufHandle
;           D0 --> ULONG Schlüsselwert der zu suchenden Daten (0-7 !!)
;
; Ergebnis: D0 --> BOOL - UPPER, wenn die Daten gefunden wurden
;                - EQUAL, wenn sie nicht gefunden wurden
;                - LOWER, wenn Fehler (Info mit LastError())
;
; Siehe auch: FindData(), FindPrev(), FreeFinds(), SetBookmark(),
;             GotoBookmark()
;=====
```

1.29 *** extra.library / FindPrev() (OFFSET -156) V1.323 ***

```
;=====
; Diese Funktion sucht das vorige Auftreten der Daten eines vorangegangenen
; FindData()-Aufrufs. Die Suche beginnt wie bei o.g. Funktion wieder an der
; aktuellen Cursorposition. Mit dem Schlüssel geben Sie an, von welchen Daten
; das vorhergehende Auftreten gesucht werden soll.
;-----
; Synopsis: success = FindPrev (BufHandle, Key)
;           D0*                A0      D0
;
; Eingaben: A0 --> APTR auf den gewünschten BufHandle
;           D0 --> ULONG Schlüsselwert der zu suchenden Daten (0-7 !!)
;
; Ergebnis: D0 --> BOOL - UPPER, wenn die Daten gefunden wurden
;                - EQUAL, wenn sie nicht gefunden wurden
;                - LOWER, wenn Fehler (Info mit LastError())
;
; Siehe auch: FindData(), FindNext(), FreeFinds(), SetBookmark(),
;             GotoBookmark()
;=====
```

1.30 *** extra.library / FreeFinds() (OFFSET -162) V1.323 ***

```

;=====
; Diese Funktion gibt die internen Kopien aller Such-Daten (alle vorhandenen
; Schlüsselwerte) wieder frei. Dies kann manchmal sehr nützlich sein, wenn
; Speichermangel besteht, und die Daten ohnehin nicht mehr benötigt werden.
;-----
; Synopsis:      success = FreeFinds (BufHandle)
;                D0*                A0
;
; Eingaben:      A0 --> APTR auf den gewünschten BufHandle
;
; Ergebnis:      D0 --> BOOL - TRUE , wenn alles in Ordnung
;                - FALSE, wenn Fehler (Info mit LastError())
;
; Bemerkung:     Nach Aufruf dieser Funktion müssen Sie erst wieder neue Daten
;                via FindData() initialisieren, bevor Sie wieder die Funk-
;                tionen für fortlaufendes bzw. rückläufiges Suchen verwenden
;                können.
;                Diese Funktion löscht automatisch die Prozessor-Caches.
;
; Siehe auch:   FindData(), FindNext(), FindPrev(), SetBookmark(),
;                GotoBookmark()
;=====

```

1.31 *** extra.library / SetBookmark() (OFFSET -168) V1.323 ***

```

;=====
; Mit dieser Funktion erklären Sie die aktuelle Cursorposition zu einer
; Bookmark. Diese Bookmarks (max. 16 pro Puffer) werden bei Speicher- oder
; Löschooperationen im Puffer automatisch Neuberechnet, und halten somit
; immer die markierte Stelle im »Gedächtnis«.
;-----
; Synopsis:      success = SetBookmark (BufHandle, Key)
;                D0*                A0      D0
;
; Eingaben:      A0 --> APTR auf den gewünschten BufHandle
;                D0 --> ULONG Nummer der zu setzenden Bookmark (0..15 !!)
;
; Ergebnis:      D0 --> BOOL - TRUE , wenn alles in Ordnung
;                - FALSE, wenn Fehler (Info mit LastError())
;
; Bemerkung:     Beachten Sie, daß jede Bookmark nur einmal vergeben werden
;                kann. Das heißt, wenn Sie eine Bookmark verwenden, die Sie
;                schon früher gesetzt haben, dann wird diese jetzt mit der
;                neuen Position überschrieben.
;
; Siehe auch:   FindData(), FindNext(), FindPrev(), FreeFinds(),
;                GotoBookmark()
;=====

```

1.32 *** extra.library / GotoBookmark() (OFFSET -174) V1.323 ***

```

;=====
; Mit dieser Funktion rufen Sie eine vorher gesetzte Bookmark ab. Das heißt,
; daß die aktuelle Cursorposition auf die in der Bookmark gespeicherten Po-
; sition gesetzt wird. Dies funktioniert natürlich nur, wenn die Daten an be-
; sagter Stelle noch existieren, d.h. sie noch nicht gelöscht wurden. Alter-
; nativ zu dieser Funktion können Sie auch SeekBuf() verwenden.
;-----
; Synopsis:      success = GotoBookmark (BufHandle, Key)
;                D0*                A0        D0
;
;
; Eingaben:      A0 --> APTR   auf den gewünschten BufHandle
;                D0 --> ULONG  Nummer der gewünschten Bookmark (0..15 !!)
;
; Ergebnis:      D0 --> BOOL  - TRUE , wenn alles in Ordnung
;                    - FALSE, wenn Fehler (Info mit GetLastError())
;
; Bemerkung:     Sollte diese Funktion fehlschlagen, dann bleibt die alte
;                Cursorposition erhalten.
;
; Siehe auch:   SeekBuf(), FindData(), FindNext(), FindPrev(),
;                FreeFinds(), SetBookmark()
;=====

```

1.33 *** extra.library / FindString() (OFFSET -180) V1.323 ***

```

;=====
; Diese Funktion sucht eine Bytefolge innerhalb einer anderen Bytefolge. Die
; Suche findet dabei »Case-Sensitiv« statt, das heißt, »A« ist ungleich »a«.
; Der verwendete Suchalgorithmus arbeitet mit »3-Phase-PreCheck«, womit so-
; gar auf alten 68000er-Amigas noch ein Datendurchsatz von ca. 130 KByte/Sec
; erreicht werden kann (je nach Auslastung des Multitaskings).
;-----
; Synopsis:      position = FindString (Source, SourceLen, Find, FindLen)
;                D0*                A0!        D0        A1!        D1
;
;
; Eingaben:      A0 --> STRPTR  auf Adresse der zu durchsuchenden Bytefolge
;                D0 --> ULONG   Länge der zu durchsuchenden Folge in Byte
;                A1 --> STRPTR  auf Adresse der zu suchenden Bytefolge
;                D1 --> ULONG   Länge der zu suchenden Folge in Byte
;
; Ergebnis:      D0 --> LONG   - die Position, ab welchem Byte der durchsuchten
;                    Bytefolge die Gesuchte in ihr enthalten ist
;                    (ACHTUNG: Zählbeginn = 1 !!)
;                    - ist NULL, wenn die gesuchte Bytefolge nicht in
;                    der Durchsuchten enthalten ist
;                    - ist negativ, wenn ein Fehler aufgetreten ist
;                    (Find-String länger als Source-String)
;
; Bemerkung:     Ich habe die Zeiger auf die Bytefolgen als STRPTR deklariert,
;                da man diese Funktion wohl meistens für die Suche innerhalb
;                von alphanumerischen Zeichenketten benutzen wird. Die Funktion
;                kann aber genauso gut zum Suchen von nicht druckbaren Zeichen
;                benutzt werden, d.h. sie bricht ihre Arbeit nicht etwa beim
;                Auftauchen eines 0-Bytes o.ä. ab. Um dies deutlich zu machen,
;
;=====

```

```

;         rede ich hier auch absichtlich immerzu nur von »Bytefolgen«,
;         und nicht etwa von »Strings« oder »Zeichenketten«.
;
; Siehe auch:  GetLength()
;=====

```

1.34 *** extra.library / GetLength() (OFFSET -186) V1.323 ***

```

;=====
; Diese Funktion ermittelt die Länge einer Bytefolge, welche mit dem spezi-
; fizierten Byte abgeschlossen ist. Man kann so z.B. die Länge eines 0-ter-
; minierten Strings ermitteln oder in Textpuffern die Linefeeds suchen und
; somit die Länge einer Zeile ermitteln etc..
;-----
; Synopsis:      length = GetLength (Start, EndByte, MaxBytes)
;                D0*                A0!         D0         D1
;
; Eingaben:      A0 --> STRPTR auf die Startadresse der Bytefolge
;                D0 --> UBYTE  zu suchendes EndByte (0-255)
;                D1 --> ULONG  max. zu prüfende Byteanzahl (s. Bemerkung)
;
; Ergebnis:      D0 --> LONG  - Länge der Bytefolge (ohne EndByte)
;                - negativ, wenn innerhalb eines angegebenen Be-
;                reichs kein EndByte gefunden wurde
;
; Bemerkung:     Der Parameter »MaxBytes« ist ein Grenzwert. Wird er erreicht,
;                dann wird die Prüfung der Bytefolge abgebrochen. Dies erweist
;                sich als nützlich, wenn innerhalb eines Puffers gesucht wird,
;                dieser jedoch nicht überschritten werden soll, wenn bis zu
;                dessen Ende kein EndByte gefunden wurde. Soll jedoch wirklich
;                bis zum Auftauchen des Endwertes gesucht werden, dann geben
;                Sie für diesen Parameter einfach -1 an.
;                Ich habe den Zeiger auf die Bytefolge als STRPTR deklariert,
;                da man diese Funktion wohl meistens für die Längenbestimmung
;                von alphanumerischen Zeichenketten benutzen wird. Die Funktion
;                kann aber genauso gut zur Längenbestimmung eines Datenpuffers
;                benutzt werden, d.h. sie bricht ihre Arbeit nicht etwa beim
;                Auftauchen eines 0-Bytes o.ä. ab (es sei denn, dieses Byte ist
;                als »EndByte« spezifiziert). Um dies deutlich zu machen, rede
;                ich hier auch absichtlich immerzu nur von einer »Bytefolge«,
;                und nicht etwa von einem »String« oder einer »Zeichenkette«.
;
; Siehe auch:    FindString()
;=====

```

1.35 *** extra.library / LastError() (OFFSET -192) V1.323 ***

```

;=====
; Ähnlich der Funktion »IoErr()« der dos.library ermittelt diese Funktion
; die Nummer des zuletzt aufgetretenen »extra.library«-Fehlers innerhalb des
; jeweiligen Tasks, der diese Funktion aufruft.
;-----

```

```
; Synopsis:      error = LastError (VOID)
;                D0
;
; Ergebnis:      D0 --> ULONG Fehlernummer (s. extradefs.i)
;=====
```

1.36 *** extra.library / LongToString() (OFFSET -198) V1.323 ***

```
;=====
; Mit dieser Funktion können Sie eine 32-Bit-Binärzahl (LONG) in einen druck-
; baren Dezimalstring umwandeln. Die Zahl wird dabei entweder als vorzeichen-
; los od. vorzeichenbehaftet interpretiert. Der String wird mit einem 0-Byte
; abgeschlossen. Als Ergebnis erhalten Sie den Adress-Zeiger auf den String.
;-----
; Synopsis:      string = LongToString (Value, Flag)
;                D0                D0      D1
;
; Eingaben:      D0 --> LONG die umzuwandelnde Zahl
;                D1 --> BOOL - TRUE  -> vorzeichenbehaftet
;                - FALSE -> vorzeichenlos
;
; Ergebnis:      D0 --> STRPTR auf den generierten Dezimalstring(+0)
;
; Bemerkung:     Sie brauchen sich nicht um die Beschaffung bzw. Freigabe des
;                Stringpuffers zu kümmern, da der benötigte Speicherplatz Ihrem
;                Task beim Öffnen der »extra.library« zur Verfügung gestellt
;                wird, und dann beim Schließen der Library auch automatisch
;                wieder freigegeben wird. Beachten Sie jedoch, daß der Puffer
;                aufgrund dieser Verwaltungsmethode beim jeweils nächsten Auf-
;                ruf dieser Funktion wieder überschrieben wird.
;
; ACHTUNG:       Benutzen Sie IMMER den Rückgelieferten Wert als Startadresse
;                ----- zum Auslesen des Strings. Der Puffer hat zwar seine feste Po-
;                sition, aber da der String rechtsbündig in diesem liegt, und
;                der String selbst ja bei jedem Aufruf eine andere Länge haben
;                kann, ändert sich damit natürlich auch die logischer Weise
;                links liegende Startadresse.
;
; Siehe auch:    StringToLong()
```

1.37 *** extra.library / StringToLong() (OFFSET -204) V1.323 ***

```
;=====
; Diese Funktion wandelt einen Dezimalstring, soweit möglich, in eine 32-Bit-
; Binärzahl (LONG) um. Der Dezimalstring darf außer den Ziffern 0-9 an füh-
; render Stelle auch ein Minuszeichen sowie an beliebiger Stelle einen Dezi-
; malpunkt enthalten. Die Umwandlung erfolgt immer Vorzeichenbehaftet.
;-----
; Synopsis:      value = StringToLong (DezString, Len)
;                D0                A0      D0
;
;=====
```

```

; Eingaben:      A0 --> STRPTR auf den umzuwandelnden Dezimalstring
;                (-2147483648 bis 2147483647)
;
;                D0 --> ULONG die Länge des Dezimalstrings
;
; Ergebnis:      D0 --> LONG die durch den String repräsentierte 32-Bit-Zahl
;
; Bemerkung:     Wenn Sie einen Dezimalpunkt im String angeben, dann wird nur
;                der Vorkommawert (INTEGER) umgewandelt.
;
; ACHTUNG:       Da das Ergebnis ja stets jeden beliebigen Wert annehmen kann,
;                ist es nicht möglich, eine generelle Fehlermeldung über das
;                Rückgabe-Register auszugeben. Wenn Sie also nicht garantieren
;                können, daß es sich um einen auswertbaren Dezimalstring
;                handelte, dann sollten Sie im Anschluß an diese Funktion immer
;                SetLastError() aufrufen, um festzustellen, ob das Ergebnis
;                gültig ist, oder ob ein Fehler auftrat. (s.a. extradefs.i)
;
; Siehe auch:    LongToString()
;=====

```

1.38 *** extra.library / PackByteRun1() (OFFSET -210) V1.323 ***

```

;=====
; Diese Funktion führt an dem übergebenen Datenbereich eine cmpByteRun1-
; Kompression durch. Dies ist die Standardmethode, mit der IFF-ILBM-Grafiken
; komprimiert werden. Für andere Daten als Grafiken ist diese Methode auch
; wenig geeignet, und wird bei solchen daher auch kaum Ersparnis bringen.
;-----
; Synopsis:      outLen = PackByteRun1 (InBuf, InLen, OutBuf, OutLen)
;                D0*                A0!      D0      A1!      D1!
;
; Eingaben:      A0 --> APTR auf die Quelldaten (zu komprimierende Daten)
;                D0 --> ULONG Anzahl zu verarbeitender Bytes der Quelldaten
;                A1 --> APTR auf den Ausgabepuffer für die komprimierten Daten
;                D1 --> ULONG Größe des Ausgabepuffers in Byte
;
; Ergebnis:      D0 --> LONG - Anzahl der geschriebenen Bytes im Ausgabepuffer
;                - negativ, wenn Ausgabepuffer zu klein
;
; Siehe auch:    UnpackByteRun1()
;=====

```

1.39 *** extra.library / UnpackByteRun1() (OFFSET -216) V1.323 ***

```

;=====
; Diese Funktion ist das Gegenstück zu PackByteRun1(). Sie entpackt die
; von o.g. Funktion komprimierten Daten wieder zurück ins Originalformat.
;-----
; Synopsis:      count = UnpackByteRun1 (InParam, OutBuf, OutLen)
;                D0                A0!      A1!      D0
;
; Eingaben:      A0 --> APTR auf eine UnpackParam-Struktur (s. extradefs.i)

```

```

;           A1 --> APTR  auf den Ausgabepuffer für die entpackten Daten
;           D0 --> ULONG Größe des Ausgabepuffers in Bytes (= Anzahl zu
;                   entpackender Bytes)
;
; Ergebnis:   D0 --> ULONG Anzahl tatsächlich entpackter Bytes, ist dieser
;                   Wert kleiner als die gewünschte Anzahl, dann sind
;                   alle Daten entpackt.
;
; Bemerkung:  Die Parameter-Struktur muß jeweils beim ersten Aufruf für ei-
;                   nen gepackten Datenblock initialisiert werden, und dient dann
;                   bei weiteren Aufrufen als Kennung des jeweiligen Datenblocks.
;                   Auf diese Weise können Sie einen größeren Datenblock in mehre-
;                   ren kleinen Einheiten wieder entpacken. Sie brauchen dann nur
;                   immer den Ausgabepuffer und dessen Größe neu angeben und die-
;                   selbe Struktur verwenden, bis alle Daten entpackt sind.
;                   Diese Funktion löscht automatisch die Prozessor-Caches.
;
; ACHTUNG:    In C/C++ ist die A0-APTR-Eingabe als Typ STRUCT UnpackParam*
;             ----- zu behandeln.
;
; Siehe auch: PackByteRun1()
;=====

```

1.40 *** extra.library / PrintIoError() (OFFSET -222) V1.323 ***

```

;=====
; Ist bei einer DOS-Operation ein Fehler aufgetreten, dann rufen Sie die
; Funktion »IoErr()« der dos.library auf. Anschließend, wenn Sie eventuell
; schon eine Fehlerbearbeitung vorgenommen haben, können Sie die Fehler-
; nummer an diese Funktion weiterleiten. Es wird die verbale Fehlermeldung
; zu dieser Nummer ermittelt, und dann die ganze Sache ordentlich mittels
; ShowMsg() präsentiert. Sie brauchen dann nur noch darauf zu reagieren,
; ob der Anwender einen nochmaligen Versuch wünscht oder nicht.
;-----
; Synopsis:   button = PrintIoError (DOSErrNum)
;             D0*                D0
;
; Eingaben:   D0 --> ULONG ein beliebiger positiver Wert (meist ein DOS-Error)
;
; Ergebnis:   D0 --> BOOL - UPPER, wenn RETRY (linke Maustaste)
;                   - EQUAL, wenn ABORT (rechte Maustaste)
;                   - LOWER, wenn Fehler (Info mit GetLastError())
;
; Bemerkung:  Wenn der angegebene Wert keine bekannte Fehlernummer ist, so
;             wird die Meldung »UNKNOWN_ERROR_NUMBER« verwendet.
;=====

```

1.41 *** extra.library / FlipCase() (OFFSET -228) V1.323 ***

```

;=====
; Diese Funktion wandelt alle Kleinbuchstaben des angegebenen Strings in ihre
; entsprechenden Großbuchstaben um, oder umgekehrt. Bei dieser Operation wer-

```

```
; den auch Umlaute und internationale Sonderzeichen korrekt berücksichtigt,
; sofern diese im normalen ASCII-Code, und nicht in Latin-Codesets o.ä. ver-
; kerksten Standards vorliegen.
```

```
-----
; Synopsis:    VOID FlipCase (String, Len, Flag)
;              A0!    D0    D1
;
; Eingaben:    A0 --> STRPTR auf den umzuwandelnden String
;              D0 --> ULONG  die Länge des Strings
;              D1 --> BOOL  - TRUE   für klein -> groß
;              - FALSE für groß  -> klein
=====
```

1.42 *** extra.library / CmpMem() (OFFSET -234) V1.323 ***

```
-----
; Diese Funktion vergleicht zwei Speicherbereiche miteinander. Es wird auto-
; matisch die schnellste Methode für den Vergleich gewählt.
```

```
-----
; Synopsis:    result = CmpMem (BlockA, BlockB, Size)
;              D0*           A0    A1    D0
;
; Eingaben:    A0 --> APTR  auf die Startadresse von Bereich #1
;              A1 --> APTR  auf die Startadresse von Bereich #2
;              D0 --> ULONG Anzahl der zu prüfenden Bytes
;
; Ergebnis:    D0 --> ULONG - ist NULL, wenn die Bereiche identisch sind
;              - die Position der ersten Abweichung, wenn nicht
;              identisch (ACHTUNG: Zählbeginn = 1 !!)
=====
```

1.43 *** extra.library / CopyMemI() (OFFSET -240) V1.323 ***

```
-----
; Hierbei handelt es sich um eine intelligente Speicher-Kopierfunktion.
; Intelligent in sofern, als daß sie automatisch erkennt, ob sich der Quell-
; und der Zielbereich an irgendeiner Stelle überlappen. Ist dies der Fall,
; wird ermittelt, ob mit aufwärts oder abwärts zählenden Adressen kopiert
; werden muß, um ein überschreiben von noch nicht kopierten Quellabschnitten
; zu vermeiden. Es wird der schnellste Weg für den Kopiervorgang gewählt.
```

```
-----
; Synopsis:    VOID CopyMemI (Source, Dest, Size)
;              A0    A1!    D0
;
; Eingaben:    A0 --> APTR  auf die Startadresse des Quell-Datenbereichs
;              A1 --> APTR  auf die Startadresse des Ziel-Datenbereichs
;              D0 --> ULONG Anzahl der zu kopierenden Bytes
;
; Bemerkung:   Diese Funktion löscht automatisch die Prozessor-Caches.
=====
```

1.44 * extra.library / FillMem() (OFFSET -246) V1.323 *****

```

;=====
; Diese Funktion füllt den angegebenen Speicherbereich mit dem spezifizierten
; Wert. Anhand der Größe des Wertes wird ermittelt, ob eine Byte-, Wort- oder
; Langwort-Füllung durchgeführt werden muß. Es wird der schnellste Weg für
; die Füllung gewählt. Der Füllwert wird immer als Vorzeichenlos betrachtet.
;-----
; Synopsis:      success = FillMem (Start, Size, Value)
;                D0*                A0!      D0      D1
;
; Eingaben:      A0 --> APTR   auf den zu füllenden Bereich
;                (ACHTUNG: gerade Adresse bei WORD/LONG-Füllung)
;                D0 --> ULONG  Größe des Bereichs in Bytes !!
;                D1 --> ULONG  Füllwert (BYTE: 0-255, WORD: >255, LONG: >65535)
;
; Ergebnis:      D0 --> BOOL  - TRUE , wenn alles in Ordnung
;                - FALSE, wenn Fehler (Info mit GetLastError())
;
; Bemerkung:     Bei Wort- und Langwort-Modus wird die Größe auf das nächste
;                Vielfache von 2 bzw. 4 abgerundet, um so eine Bereichsüber-
;                schreitung zu vermeiden. Achten Sie darauf, daß der Bereich
;                mindestens so groß ist, um einen Wert aufzunehmen, anderen-
;                falls wird aufgrund der Abrundung keine Füllung durchgeführt.
;                Diese Funktion löscht automatisch die Prozessor-Caches.
;=====

```

1.45 * extra.library / GetMem() (OFFSET -252) V1.323 *****

```

;=====
; Diese Funktion belegt Speicher via »AllocMem()« der exec.library, bindet
; aber zusätzlich sämtliche Bereiche in die Speicherliste der Task-Struktur
; des aufrufenden Tasks ein. Somit wird verhindert, daß der Speicherplatz bis
; zum nächsten Reset blockiert bleibt, wenn vergessen wurde, diesen wieder
; freizugeben, bevor das Programm beendet wurde. In diesem Falle wird der be-
; legte Speicherplatz automatisch wieder freigegeben, wenn der Task aus dem
; System entfernt wird.
;-----
; Synopsis:      address = GetMem (Size, Requirements)
;                D0*                D0      D1
;
; Eingaben:      D0 --> ULONG  die Größe des gewünschten Bereichs in Byte
;                D1 --> ULONG  die Eigenschaften des gewünschten Bereichs
;
; Ergebnis:      D0 --> APTR  - auf die Adresse des belegten Bereichs
;                - 0-PTR, wenn die gewünschte Speicherart bzw.
;                Größe nicht verfügbar ist
;
; Bemerkung:     Der Wert MEMF_CLEAR zum Löschen des belegten Bereichs wird
;                intern mit den Requirements verknüpft und braucht somit nicht
;                explizit angegeben werden.
;
; Siehe auch:    UnGetMem()
;=====

```

1.46 * extra.library / UnGetMem() (OFFSET -258) V1.323 *****

```

;=====
; Diese Funktion ist das Gegenstück zu GetMem(). Sie gibt entweder den an-
; gegebenen Speicherblock oder alle Blöcke, welche mittels der o.g. Funktion
; für den aufrufenden Task belegt wurden, wieder frei.
;-----
; Synopsis:      VOID UnGetMem (Block)
;                  A0
;
; Eingaben:      A0 --> APTR - die Adresse eines Speicherblocks von GetMem()
;                  - ein 0-PTR gibt alle Speicherblöcke frei
;
; Siehe auch:   GetMem()
;=====

```

1.47 * extra.library / OutlineOff() (OFFSET -264) V1.323 *****

```

;=====
; Diese Funktion schaltet den AreaOutline-Mode des angegebenen RastPorts aus.
;-----
; Synopsis:      VOID OutlineOff (RastPort)
;                  A0!
;
; Eingaben:      A0 --> APTR auf eine gültige RastPort-Struktur
;
; ACHTUNG:      In C/C++ ist die A0-APTR-Eingabe als Typ STRUCT RastPort*
;                zu behandeln.
;
; Siehe auch:   OutlineOn(), SetOPen()
;=====

```

1.48 * extra.library / OutlineOn() (OFFSET -270) V1.323 *****

```

;=====
; Diese Funktion schaltet den AreaOutline-Mode des angegebenen RastPorts ein.
;-----
; Synopsis:      VOID OutlineOn (RastPort)
;                  A0!
;
; Eingaben:      A0 --> APTR auf eine gültige RastPort-Struktur
;
; ACHTUNG:      In C/C++ ist die A0-APTR-Eingabe als Typ STRUCT RastPort*
;                zu behandeln.
;
; Siehe auch:   OutlineOff(), SetOPen()
;=====

```

1.49 * extra.library / SetOPen() (OFFSET -276) V1.323 *****

```

;=====
; Diese Funktion setzt den für die Area-Outlines zu benutzenden Farbstift.
;-----
; Synopsis:      VOID SetOPen (RastPort, Pen)
;                A0!      D0!
;
;
; Eingaben:      A0 --> APTR   auf eine gültige RastPort-Struktur
;                D0 --> UBYTE  Nummer des zu verwendenden Farbstiftes
;
; ACHTUNG:      Diese Funktion schaltet den AreaOutline-Modus automatisch
; -----
;                ein, wenn er noch nicht aktiv sein sollte !!
;                In C/C++ ist die A0-APTR-Eingabe als Typ STRUCT RastPort*
;                zu behandeln.
;
; Siehe auch:   OutlineOff(), OutlineOn()
;=====

```

1.50 *** extra.library / SpecText() (OFFSET -282) V1.323 ***

```

;=====
; Diese Funktion arbeitet wie »Text()« der graphics.library, nur wird hier
; um den Text eine Umrandung gezeichnet, wodurch Sie bei entsprechender Farb-
; wahl u.a. Schatten-, Outline- od. Reliefeffekte erzeugen können. Durch
; Flags haben Sie auch die Möglichkeit z.B. die aktuellen Farben eines Rast-
; ports zu verwenden, was es ermöglicht die Farben in höheren Programmier-
; sprachen einfach durch Verwendung der COLOR-Befehle solcher Sprachen ein-
; zustellen bzw. zu verändern. Ein weiterer Unterschied zu der Gfx-Funktion
; ist der, daß sich die X/Y-Positionen hier auf die linke obere Ecke be-
; ziehen, und nicht wie bei »Text()« auf die Baseline des verwendeten Fonts.
;-----
; Synopsis:      success = SpecText (RastPort, String, xPos, yPos)
;                D0*                A0!      A1!      D0!      D1!
;
;
; Eingaben:      A0 --> APTR   auf eine gültige RastPort-Struktur
;                A1 --> STRPTR  auf den auszugebenden Text(+0)
;                D0 --> UWORD  die X-Koordinate, wo der Text erscheinen soll
;                D1 --> UWORD  die Y-Koordinate, wo der Text erscheinen soll
;
; Ergebnis:      D0 --> BOOL - TRUE , wenn alles funktioniert hat
;                - FALSE, wenn zu wenig Speicher für benötigte
;                IntuiText-Strukturen
;
; Bemerkung:     Bevor Sie diese Funktion aufrufen sollten Sie mit der Funk-
;                tion SpecTextPrefs() einstellen, wie diese Funktion arbei-
;                ten soll. Wenn keine Einstellungen vorgenommen wurden, dann
;                werden folgende Defaults benutzt: MPen=0, TLPen=2, BRPen=1,
;                Style=STS_Plain, Maske=STM_Outline und Flags=STF_AP2MP. Die
;                Einstellungen des angegebenen RastPorts werden durch diese
;                Funktion NICHT verändert.
;
; ACHTUNG:      Wenn der angegebene RastPort der eines Intuition-Window ist,
; -----
;                dann werden zu lange Texte am Rand autom. abgeschnitten. Han-
;                delt es sich jedoch um einen selbst eingerichteten RastPort
;                der nicht über einen Layer verfügt, dann haben Sie die Verant-
;
;
;=====

```

```

;          worgung sicherzustellen, daß der Rand keinesfalls überschrie-
;          ben wird (X/Y u. Textlänge beachten). Anderenfalls könnte es
;          zu einem Systemabsturz mit Datenverlust kommen, für den ich
;          KEINERLEI HAFTUNG übernehme !!
;          In C/C++ ist die A0-APTR-Eingabe als Typ STRUCT RastPort*
;          zu behandeln.
;
; Siehe auch: SpecTextPrefs()
;=====

```

1.51 *** extra.library / SpecTextPrefs() (OFFSET -288) V1.323 ***

```

;=====
; Mit dieser Funktion bestimmen Sie, wie die Funktion SpecText() dieser
; Library arbeiten soll. Die Einstellungen gelten nur für den aufrufenden
; Task, jeder Task kann also seine eigenen Prefs verwenden, ohne daß sie
; sich gegenseitig in die Quere kommen. (möchte ja wohl auch sein, oder ??)
;-----
; Synopsis:   VOID SpecTextPrefs (MPen, TLPen, BRPen, Style, Mask, Flags)
;
;              D0!   D1!   D2!   D3!   D4!   D5!
;
; Eingaben:   D0 --> UBYTE Nr. der gewünschten Schriftfarbe (MainPen)
;             D1 --> UBYTE Nr. der Farbe für ob./li.-Umrandung (TopLeftPen)
;             D2 --> UBYTE Nr. der Farbe für un./re.-Ränder (BottomRightPen)
;             D3 --> UBYTE der zu verwendende SoftStyle (s.extradays.i)
;             D4 --> UBYTE eine Maske, die angibt welche Umrandungsteile tat-
;             sächlich gezeichnet werden sollen (s. Bemerkung)
;             D5 --> UWORD zusätzliche Spezifikationen (s. extradays.i)
;
; Bemerkung:  Wenn die Flags »STF_AP2MP«, »STF_BP2TL« bzw. »STF_OP2BR« an-
;             gegeben werden, dann werden die Parameter »MPen«, »TLPen« und
;             »BRPen« deshalb NICHT ignoriert, vielmehr werden sie vermerkt
;             und kommen zur Anwendung, wenn durch eine Änderung der Flags
;             die o.g. Flags nicht mehr länger existent sind. In der Maske
;             bedeutet jedes gesetzte Bit, daß der entsprechende Teil der
;             Umrandung gezeichnet wird. Einige vordefinierte Werte finden
;             Sie in dem Includefile »extradays.i«
;
;             Die Maske hat folgendes Format:
;             Bit-Nr. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0
;             -----+-----+-----+-----+-----+-----+-----+-----
;             Randpos.| TL | T | TR | R | BR | B | BL | L
;
;             TL=oben/links,  T=oben,  TR=oben/rechts, R=rechts,
;             BR=unten/rechts, B=unten, BL=unten/links, L=links
;
; ACHTUNG:   Wenn Sie keine Flags zur Uebernahme von nur einzelnen Kompo-
;             nenten verwenden, dann werden alle Werte übernommen.
;
; Siehe auch: SpecText()
;=====

```

1.52 *** extra.library / ShowMsg() (OFFSET -294) V1.323 ***

```

;=====
; Diese Funktion stellt eine Alternative zu der Routine »DisplayAlert()« der
; intuition.library dar. Diese Funktion arbeitet grundsätzlich genauso wie
; die Intuition-Routine, bietet aber bei der Gestaltung des Alert-Textes noch
; die Möglichkeit, Schriftfarbe und Schriftstil Ihren Wünschen entsprechend
; zu wählen. Außerdem paßt sich diese Funktion ab OS 2.0 automatisch an die
; aktuellen »overscan.prefs« an, und landet so nicht, wie ihrer großen Intu-
; ition-Schwester schon öfter passiert, außerhalb der Bildröhre ?!
;-----
; Synopsis:      button = ShowMsg (Param, Type, Height, ColorTab)
;                D0*           A0!   D0!       D1       D2!
;
; Eingaben:      A0 --> STRPTR auf den Parameter-String (s. Bemerkung)
;                D0 --> LONG   Alert-Typ, hiervon ist nur das MSB interessant:
;                        MSB (Bit 31) gesetzt DeadEnd, sonst Recovery
;                        Nach einem DeadEnd-Alert folgt ein System-Reset.
;                D1 --> ULONG  die Höhe des Alerts in Videozeilen (max. 200)
;                D2 --> APTR - auf eine optionale Farbtabelle (s. Bemerkung)
;                        - 0-PTR zur Verwendung der Standardfarben
;
; Ergebnis:      D0 --> BOOL - UPPER, wenn linke Maustaste gedrückt
;                - EQUAL, wenn rechte Maustaste gedrückt
;                - LOWER, Fehler (genaue Info mit LastError())
;
; Bemerkung:     Wenn Ihnen die Standardfarben nicht recht sind, dann können
;                Sie sie durch die Angabe einer Farbtabelle folgenden Formats
;                auf die von Ihnen gewünschten Werte ändern. Für alle Farben,
;                für die kein Eintrag in der Tabelle existiert, werden weiter-
;                hin die Standardfarben benutzt. Die Farbanteile können jeweils
;                Werte von 0-15 annehmen. Assembler-Programmierer können zur
;                einfacheren Erstellung der Tabelle das Macro SMCOLOR() aus
;                dem Includefile »extradefs.i« verwenden.
;                Für BASIC u. C/C++ sind hierfür ähnliche Funktionen bzw.
;                Macros in den entsprechenden Includefiles definiert. Bitte
;                entnehmen Sie die Beschreibung jener Macros bitte direkt aus
;                den entsprechenden Includefiles.
;
;                Aufbau der Farbtabelle:
;                +-->  UBYTE Nr. der Farbe (3-15) (0-2 werden ignoriert)
;                |    + UWORD Rotanteil*256 + Grünanteil*16 + Blauanteil
;                |    + UBYTE folgt ein weiterer Farbeintrag ?? (0=nein/1=ja)
;                |
;                |    wenn weiterer Farbeintrag=ja, dann --+
;                |
;                +-----+
;
;                Aufbau des Parameter-Stings:
;                +-->  UWORD X-Koordinate dieses Teilstrings ----\ (jeweils bezogen
;                |    + UBYTE Y-Koordinate dieses Teilstrings ----/ auf li./ob. Ecke)
;                |    + UBYTE neue SpecTextPrefs ?? (0=nein/1=ja)
;                |
;                |    wenn SpecTextPrefs=ja, dann
;                |    + UBYTE Nr. des für SpecText() zu verwendenden »MPen« \
;                |    + UBYTE Nr. des zu verwendenden »TLPen« _____/

```

```

; | + UBYTE Nr. des zu verwendenden »BRPen« / \ /
; | + UBYTE der zu verwendende SoftStyle _/ \ /
; | + UBYTE die zu verwendende Randmaske _/(s. SpecTextPrefs())
; | ansonsten folgt direkt
; |
; | + TEXT der auszugebende Text selbst mit abschließendem 0-Byte
; | + UBYTE folgt ein weiterer Teilstring ?? (0=nein/1=ja)
; |
; | wenn weiterer Teilstring=ja, dann --+
; |
; +-----+
;
; - Zur einfacheren Erstellung des Parameter-Strings können
; Assembler-Programmierer die Macros SMTEXTP() u. SMTEXT()
; aus dem mitgelieferten Includefile »extradefs.i« verwenden.
; Für BASIC u. C/C++ sind hierfür ähnliche Funktionen bzw.
; Macros in den entsprechenden Includefiles definiert. Bitte
; entnehmen Sie die Beschreibung jener Macros bitte direkt aus
; den entsprechenden Includefiles.
;
; Tabelle der vorhandenen Standardfarben:
; 1 - hellgrau (nur für TLPen u. BRPen) 3 - rot(100%)
; 4 - grün(100%) 5 - blau(100%) 6 - gelb(100%)
; 7 - violett(100%) 8 - türkis(100%) 9 - weiß(100%)
; 10 - weinrot 11 - orange 12 - pink
; 13 - schilfgrün 14 - lila 15 - himmelblau
;
; Die Farben 0-2 sind reserviert und werden automatisch in 3, 4
; bzw. 5 umgewandelt, falls sie angegeben werden. Nur für TLPen
; und BRPen ist zusätzlich auch Farbe 1 erlaubt. Als Styles und
; Masken gelten die gleichen, wie bei SpecTextPrefs()
;
; Bis zum Auftauchen der ersten Prefs im Parameterstring werden
; folgende Defaults verwendet: MPen=7, alle anderen Einträge 0.
;
; Die Erstellung des Textes erfolgt, wie Sie sicher schon er-
; raten haben, via SpecText() dieser Library. Allerdings wer-
; den die aktuellen Prefs Ihres Task zu genannter Funktion durch
; die hier angegebenen Prefs in keinster Weise verändert.
;
; ACHTUNG: Diese Funktion darf NUR von vollwertigen DOS-Prozessen auf-
; ----- gerufen werden. Ein normaler EXEC-Task ist nicht ausreichend.
;
; Nur OCS/ECS/AGA-Version:
; -----
;
; Der von dieser Funktion benutzte RastPort hat KEINEN Layer,
; d.h. Sie müssen selbst aufpassen, daß die Texte innerhalb des
; RastPorts bleiben (640 x angegebene Höhe). Durch die Verwen-
; dung von 16 Farben benötigt diese Funktion gegenüber der In-
; tuition-Routine wesentlich mehr ChipRam (bei max. Höhe 64 KB).
; Sollte diese Menge nicht mehr verfügbar sein, so bricht diese
; Funktion mit einer entsprechenden Fehlermeldung ab.
;
; Nur CGX-Version:
; -----
;
; Bei der CGX-Version kann es vorkommen, das der Mauszeiger nach
; abschalten des Sprite-DMA (während der Anzeige der Message)

```

```

; weiterhin sichtbar bleibt. Das hängt davon ab, ob die GfxBase
; gültige SimpleSprite-Einträge für alle Hardware-Sprites hat.
; Wenn Sie grundsätzlich eine CGX-Workbench verwenden, und Ihr
; System hochfahren ohne vor Öffnung des WB-Screens irgendwelche
; Ausgaben durch die Startup-Sequenz zu tätigen, dann sind o.g.
; Einträge in der GfxBase immer uninitialized, und der Maus-
; zeiger bleibt demzufolge immer sichtbar. Um dies zu ändern, muß
; einfach mal ein normaler Amiga-Screen (PAL, NTSC, EURO36 etc.)
; geöffnet werden, was dazu führt, daß die Einträge initialisiert
; werden. Eine zweite Möglichkeit ist, einfach die ENV-Variable
; KEEPAMIGAVIDEO von CyberGraphX zu aktivieren.
;
; Siehe auch: SpecText(), SpecTextPrefs()
;=====

```

1.53 *** extra.library / SMCOLOR() ***

```

;=====
; Dieses Macro erstellt einen Farbeintrag für die optional anzugebende Farb-
; tabelle für die Funktion ShowMsg().
;-----
; Definiert:   extradefs.i   (Copyright © 1998-2001 RhoSigma, Roland Heyder)
;
; Schablone:   Pen/A/N, Red/A/N, Green/A/N, Blue/A/N, END/K/S
;
; Parameter:   Pen    --> Nr. des zu ändernden Farbstiftes (3-15)
;              (0-2 sind reserviert u. werden ignoriert)
;
;              Red    --> Rot-Anteil der neuen Farbe (0-15)
;              Green  --> Grün-Anteil der neuen Farbe (0-15)
;              Blue   --> Blau-Anteil der neuen Farbe (0-15)
;
;              END    --> Wird dieses Schlüsselwort angegeben, dann endet die
;              Farbtabelle an dieser Stelle, anderenfalls muß ein
;              weiterer Farbeintrag (Macro-Aufruf) folgen.
;
; Bemerkung:   Ein Beispiel für die Benutzung dieses Macros finden Sie in der
;              Datei »demo«/»ShowMsgDemo.ASM«.
;
; Siehe auch:  SMTEXTP(), SMTEXT(), ShowMsg()
;=====

```

1.54 *** extra.library / SMTEXTP() ***

```

;=====
; Dieses Macro erstellt einen Teilstring des Parameterstrings der Funktion
; ShowMsg() inclusive der gewünschten Einstellungen für die Textausgabe.
; Wahlweise kann der Text automatisch zentriert, oder links- bzw. rechtsbündig
; ausgerichtet werden.
;-----
; Definiert:   extradefs.i   (Copyright © 1998-2001 RhoSigma, Roland Heyder)
;

```

```

; Schablone:   xPos/A/N, yPos/A/N, MPen/A/N, TLPen/A/N, BRPen/A/N, Style/A/N,
;             Mask/A/N, Text/A, END/K/S
;
; Parameter:  xPos --> x-Position des Textes,
;             oder einer der oben definierten SMTPOS-Werte
;
;             yPos --> y-Position des Textes,
;             Ausrichtung erfolgt an der Oberkante des Fonts
;
;             MPen --> gewünschter MainPen          \
;             TLPen --> gewünschter TopLeftPen       |
;             BRPen --> gewünschter BottomRightPen  | (s. SpecTextPrefs())
;             Style --> gewünschter SoftStyle       |
;             Mask  --> gewünschte Randmaske        /
;
;             Text --> der auszugebende Text selbst,
;             in <..>, wenn Leerzeichen enthalten sind
;
;             END  --> Wird dieses Schlüsselwort angegeben, dann endet der
;             Parameterstring an dieser Stelle, anderenfalls muß
;             ein weiterer Teilstring (Macro-Aufruf) folgen.
;
; Bemerkung:  Dieses Macro sollten Sie nur dann verwenden, wenn Sie für
;             diesen Teilstring neue Einstellungen für Farben etc. machen
;             wollen. Wenn dieser Teilstring jedoch mit den gleichen Ein-
;             stellungen ausgegeben werden soll, wie der vorhergehende bzw.
;             Sie die Default-Einstellungen verwenden wollen, dann müssen
;             Sie stattdessen das Macro SMTEXT() verwenden.
;
;             Ein Beispiel für die Benutzung dieses Macros finden Sie in der
;             Datei »demo«/»ShowMsgDemo.ASM«.
;
; Siehe auch:  SMCOLOR(), SMTEXT(), ShowMsg()
; =====

```

1.55 *** extra.library / SMTEXT() ***

```

; =====
; Dieses Macro erstellt einen Teilstring des Parameterstrings der Funktion
; ShowMsg(). Wahlweise kann der Text automatisch zentriert, oder links- bzw.
; rechtsbündig ausgerichtet werden.
; -----
; Definiert:   extradefs.i (Copyright © 1998-2001 RhoSigma, Roland Heyder)
;
; Schablone:  xPos/A/N, yPos/A/N, Text/A, END/K/S
;
; Parameter:  xPos --> x-Position des Textes,
;             oder einer der oben definierten SMTPOS-Werte
;
;             yPos --> y-Position des Textes,
;             Ausrichtung erfolgt an der Oberkante des Fonts
;
;             Text --> der auszugebende Text selbst,
;             in <..>, wenn Leerzeichen enthalten sind
;
;

```

```

;           END    --> Wird dieses Schlüsselwort angegeben, dann endet der
;                   Parameterstring an dieser Stelle, anderenfalls muß
;                   ein weiterer Teilstring (Macro-Aufruf) folgen.
;
; Bemerkung:   Dieses Macro sollten Sie nur dann verwenden, wenn Sie für
;               diesen Teilstring die gleichen Einstellungen wie für den vor-
;               hergehenden Teilstring verwenden wollen. Wenn dieser Teil-
;               string jedoch neue Einstellungen für Farben etc. erhalten
;               soll, dann müssen Sie stattdessen das Macro SMTEXTP() ver-
;               wenden.
;
;               Ein Beispiel für die Benutzung dieses Macros finden Sie in der
;               Datei »demo«/»ShowMsgDemo.ASM«.
;
; Siehe auch:  SMCOLOR(), SMTEXTP(), ShowMsg()
;=====

```

1.56 *** Ivo / Lib-Call-Macro ***

```

;=====
; Dieses Macro dient zum Aufruf einer Funktion aus der dazugehörigen Library.
; Dabei ist dieses Macro gegenüber den CALL-Macros des NDK jedoch so intelli-
; gent, daß es erkennt, ob mit ihm auch wirklich eine Funktion der zugehörigen
; Library aufgerufen wird. Damit wird vermieden, daß versehendlich z.B. eine
; Funktion der exec.library mit dem Macro (und damit natürlich auch dem Zeiger)
; der graphics.library aufgerufen werden kann, was höchstwahrscheinlich sofort
; einen Systemabsturz nach sich ziehen würde.
; Außerdem haben Optimal-Programmierer auch die Möglichkeit Quick-Aufrufe zu
; verwenden, um jedes überflüssige Byte einzusparen. Für die Programmierung re-
; entranter Programme ist ebenfalls eine Syntax definiert, bei der die voll-
; ständige effektive Adresse des Basis-Zeigers als Parameter übergeben werden
; muß, da dort ja keine flüchtigen absoluten Adressen verwendet werden dürfen.
;-----
; Definiert:    ... _lib.i (Copyright © 1998-2001 RhoSigma, Roland Heyder)
;
; Schablone:   Func/A, Opts/F/S
;
; Parameter:   Func --> Name der aufzurufenden Funktion ohne »_LVO_LIBNAME_«
;
;               Opts --> - das Schlüsselwort »OK« für einen QuickJump-Aufruf
;                           (s. Bemerkung)
;
;               - Opts weglassen, um Basis-Symbol »_LibnameBase« ins
;                 Register A6 zu laden
;
;               - die effektive Adresse, welche die Basis der Library
;                 enthält (z.B. »globals_Libname(a5)«)
;
; Bemerkung:   Wird das Schlüsselwort »OK« angegeben, dann wird davon ausge-
;               gangen, daß die richtige Basisadresse der Library bereits im
;               Register A6 steht, und es wird sofort in die Funktion gesprungen.
;=====

```