07d7b928-0

Automan@Doom.gun.de

COLLABORATORS			
	TITLE : 07d7b928-0		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Automan@Doom.gun.de	February 12, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

07d7b928-0

1.1 AmBoS - BBS.library Dokumentation

```
                 +-----------------------------------+
 |                                   |
 | AmBoS - BBS_Library Dokumentation |
 |                                   |
 | (c) 1994 Automan@Doom.gun.de      |
 |                                   |
 +-----------------------------------+


                Über~diese~Dokumentation...~~~~~~~~~~~~~~~

                Programme~unter~AmBoS,~wie~geht~das?~:-)~~
                Funktionen:


                bbs_open()
                                Externes Programm initialisieren

                bbs_close()
                               Externes Programm abschließen


Ein-/Ausgabe:


                bbs_printf()
                              Formatierte Ausgabe wie printf()

                bbs_puts()
                                Ausgabe eines Strings

                bbs_gets()
                                Eingabe eines Strings

                bbs_sgets()
                               Eingabe eines Strings (unsichtbar)

                bbs_getc()
```

```
                        Eingabe eines Zeichens

bbs_fgetc()
                       Eingabe eines Zeichens (gefiltert)

bbs_Wgetc()
                       Zeicheneingabe,Abbruch durch Signal-Bits mögl.

bbs_Wfgetc()
                      Zeicheneingabe, Signal-Bit(s), gefiltert

bbs_lookc()
User-Daten:


bbs_LoadUserData()
        User-Daten laden

bbs_SaveUserData()
        User-Daten speichern

bbs_FreeUserData()
        User-Daten freigeben


bbs_FirstUser()
           Lese User-Daten

bbs_NextUser()
            Lese weitere User-Daten


Mails und Brettinhalt:


bbs_MailToUser()
          Nachricht an einen User schicken

bbs_MailToBrett()
         Nachricht in ein Brett schreiben


bbs_GetBrettType()
        Brett-Typ ermitteln


bbs_FirstBrettInhalt()
    Brettinhalt lsen

bbs_NextBrettInhalt()
     Brettinhalt weiterlesen

bbs_SaveBrettInhalt()
     Brettinhalt speichern

bbs_FreeBrettInhalt()
     Brettinhalt freigeben
```

```
                bbs_BrettInhaltByNumber()
                 Lese bestimmten Brettinhalt


Spezielle Funktionen:


                bbs_graphic()
                             ANSI-Grafik anzeigen

                bbs_text()
                                Textfiles anzeigen

                bbs_menu()
                                Horizontales Menü erzeugen


                bbs_ObtainName()
                          String-Verwaltung

                bbs_ReleaseName()
                Include-Files:


                libraries/bbs_library.h
                   Strukturen, Definitionen zu AmBoS

                pragmas/bbs_pragmas.h
                     SAS-C Pragmas zur BBS.library

                clib/bbs_protos.h
                         C-Prototypen zur BBS.library


                bbs.fd
                                   FD-Files für andere Programmiersprachen
```

1.2 Über diese Dokumentation

```
Grundsätzliches...

Dieses ist die erste offizielle Version der "AmBoS BBS.library Dokumentation" im
AmigaGuide-Format. Sie ersetzt alle bisher in Umlauf geratenen Teil-Beschreibungen ↩
     und
ist als Referenz für alle dem Programmierer zugänglichen Funktionen der "BBS. ↩
    library"
gedacht.

Diese Dokumentation wird ständig erweitert und aktualisiert. Dennoch ist es nicht
auszuschließen, daß sich, gerade bei der ersten Version, Fehler jeder Art
eingeschlichen haben...

Der Autor übernimmt keine Haftung für Schäden, die durch Anwendung der  ↩
    beschriebenen
```

Funktionen in einer Mailbox-Umgebung entstehen können.

Diese Dokumentation ist frei kopierbar, solange keinerlei Änderungen an den im Archiv
erhaltenen Dateien vorgenommen werden.

Ich bitte alle intressierten AmBoS-User und Programmierer, mir bei der Gestaltung
dieser Dokumentation behilflich zu sein. Verbesserungsvorschläge und/oder Fehlerreports
nehme ich gerne und jederzeit per EMail entgegen (Automan@Doom.gun.de).

Ich habe mich entschlossen, keine reinen ASCII-Texte herauszugeben, da das Programm
AmigaGuide mittlerweile zum Standard geworden ist.

Was ist Was?

Zusätzlich zu diesem Dokument befinden sich im Verzeichnis "include" einige
C-Include-Files und ein FD-File zum Einbinden der Library-Funktionen in andere
Programmiersprachen:

clib/bbs_protos.h	SAS-C Prototypes
libraries/bbs_library.h	Das C-Include-File
pragma/bbs_pragmas.h	SAS_C Pragmas
fd/bbs.fd	Standard FD-File

Diese Dateien sind auch als Text in dieses Dokument aufgenommen.

Im Verzeichnis "Source" ist außerdem ein Beispielprogramm mit dem zugehörigen
C-Quellcode (SAS) zu finden. Es kann im AmBoS-Setup/Befehle direkt eingebunden werden
und demonstriert die Ausgabe eines Strings in der Mailbox-Umgebung.

Mfg. Christian (Automan)

1.3 Programme unter AmBoS - Wie geht das? :-)

Programme unter AmBoS, wie geht das ? ;-)))

Unter AmBoS haben Sie zwei Möglichkeiten eigene Programme einzubinden. Die Art ihres
Programmes bestimmt ihre Entscheidung welche der beiden Möglichkeiten für Sie die Beste
ist.

Doors -> Umlenken von Standard-Input und Standard-Output.

Diese Möglichkeit ist die wohl einfachere, da diese von Ihnen kein spezielles Wissen

erfordert. Wenn Sie schon einmal ein Programm geschrieben haben, bei dem die Ein- und Ausgabe ausschließlich im Shell-fenster erfolgt (wer hat das nicht), dann wird dieses Programm auch als AmBoS-Door verwendbar sein. Große Teile der Programmentwicklung kommen hierdurch ohne eine MailboxSoftware aus, da Sie Ihre Programme einfach in der Shell testen können. Ein weiterer Vorteil dieser Technik liegt darin, daß AmBoS nicht das einzige Mailbox-System ist, das diese Schnittstelle unterstützt, es gibt eine ganzen Menge weiterer Systeme unter denen ein so geschriebenes Programm nutzbar ist (CNet, DLG...). Der Nachteil dieser Schnittstelle ist, daß Sie keine Möglichkeit haben an box-interne Daten heranzukommen (Userdaten, Brettdaten usw.). Ein Programm als Door zu schreiben macht also vor allem für Spiele und Spielereien einen Sinn...

Externe -> Benutzen der BBS.library

Die zweite Möglichkeit wird Ihnen durch das Benutzen der BBS.library angeboten. Die Library bietet Ihnen Funktionen zur Ein- und Ausgabe, zum Scannen der User-Liste oder eines Brettinhaltes und vieles mehr. Spezielle Kenntnisse über die Funktion der seriellen Schnittstelle oder der AmBoS-Daten-Files sind nicht erforderlich. Die BBS.library wird ständig erweitert und wir warten nur auf Ihre Vorschläge für neue Funktionen ;-) . Mit Hilfe der BBS.library geschriebene Programme sind nur unter AmBoS lauffähig. Das bedeutet, Sie müssen Ihr Programm als "externes" Programm einbinden und aus AmBoS heraus starten.

AmBoS Datenfiles

Bei vielen Mailbox-Programmen ist es gang und gebe, daß die Struktur der Daten-Files offengelegt wird, sodaß jedes Programm sich die entsprechenden Daten direkt aus den Files besorgen kann, die das Boxprogramm anlegt. Dies ist unter AmBoS nicht möglich, da wir uns Änderungen in den Datenfiles in jeder Form vorbehalten und ein freier Zugriff auf die Files spätestens bei einer Mehrportbox gar nicht fehlerfrei funktionieren kann. Dies bedingt einige Einschränkungen was die zur Verfügung gestellten Daten angeht, hat aber den entscheidenden Vorteil, daß Ihre Programme mit Sicherheit auch noch mit der nächsten AmBoS-Version zusammenlaufen. Ein Teil der wichtigen Daten wird Ihnen über die BBS.library zur Verfügug gestellt, die fehlenden Daten werden wir nach und nach über

```
die Library anbieten.

Jörg Eßmann, im Dezember 1994
```

1.4 bbs_open()

```
               BBS.library/bbs_open

   NAME
        bbs_open -- Anmelden des externen Programmes am Port  (V0)

   SYNOPSIS
        ExternInfo = bbs_open(Init)
                              a1

        struct ExternInfo *bbs_open(char *)

   FUNCTION

        Meldet das externe Programm beim aufrufenden Port an.
        Diese Routine muß angesprungen werden bevor die
        anderen Routinen der Library benutzt werden können.

   INPUTS
        Init -  Der dem Programm übergebene 1. Parameter (argv[1]).
                Der Inhalt des Parameters kann je nach Boxprogramm
                unterschiedlich sein.

   RESULT
        ExternInfo - Zeiger auf die ExternInfo-Struktur. Sie enthält
                     wichtige Daten wie z.B. Username, Wohnort etc.
   BUGS

   SEE ALSO

               bbs_close()
```

1.5 bbs_close()

```
               BBS.library/bbs_close

   NAME
        bbs_close -- Abmelden des externen Programmes beim Port (V0)

   SYNOPSIS

        void bbs_close(void)

   FUNCTION
        Meldet das externe Programm beim Port ab.
        Diese Routine muß vor dem Verlassen des externen Programmes
        aufgerufen werden um die Kontrolle über den Port an die Box
```

```
     zurückzugeben.

BUGS

SEE ALSO

             bbs_open()
```

1.6 bbs_printf()

```
             BBS.library/bbs_printf

NAME
     bbs_printf

SYNOPSIS
     Zeichen bbs_printf(String, param...)
                          a1      a2

     int bbs_printf (char *,...);

FUNCTION
     Funktioniert wie printf...

BUGS

SEE ALSO

             bbs_gets()
             ,
             bbs_sgets()
             ,
             bbs_getc()
```

1.7 bbs_puts()

```
             BBS.library/bbs_puts

NAME
     bbs_puts  -- einen String ausgeben (V0)

SYNOPSIS
     bbs_puts(string)
               a1

     void bbs_puts(char *)

FUNCTION
     Schreibt einen string auf die serielle Schnittstelle, den Console-
     Screen oder auf beides.

INPUTS
```

```
     string - 0 terminierter String

BUGS

SEE ALSO

             bbs_gets()
```

1.8 bbs_gets()

```
             BBS.library/bbs_gets

NAME
     bbs_gets -- einen String einlesen

SYNOPSIS
     result bbs_gets(Deposid, MaxChars, Mode)
                     a1        d1        d2

     int bbs_gets(char *, LONG, LONG);

FUNCTION
     Liest einen String, ähnlich einem String-Gadget, ein (inclusive
     Cursorsteuerung, Delete und Backspace).
     Die durch den User eigegebenen Zeichen werden bis zur maximalen Länge
     <MaxChars> in den String <Deposid> kopiert.

     Wird in Mode eine 1 übergeben so wird ein blauer Hintergrundkasten
     in der <MaxChars> entsprechenden Länge angezeigt.

     Wird in <Deposid> ein String übergeben so wird dieser als Vorein-
     stellung angezeigt.

INPUTS
     Deposid  - Der Puffer für den zu lesenden String
     MaxChars - Die maximale Anzahl der zu lesenden Zeichen
     Mode     - Hintergrund ja oder nein

RESULT
     result   - ist NULL wenn das Externe Programm verlassen werden soll.

BUGS

SEE ALSO

             bbs_sgets()
             ,
             bbs_getc()
             ,
             bbs_fgetc()
```

1.9 bbs_sgets()

```
             BBS.library/bbs_sgets

NAME
     bbs_sgets -- (secret
             gets()
              ) liest einen nicht sichtbaren String

SYNOPSIS
     result bbs_gets(Deposid, MaxChars, Mode)
                     a1        d1        d2

     int bbs_sgets(char *, LONG, LONG);

FUNCTION
     Liest einen String ein. Die Eingabe ist dabei nicht sichtbar, alle
     Buchstaben werden durch '*' dargestellt.

     Die durch den User eigegebenen Zeichen werden bis zur maximalen
     Länge <MaxChars> in den String <Deposid> kopiert.
     (Eingabe inclusive Cursor-Steuerung, Delete und BackSpace)

     Wird in <Mode> eine 1 übergeben so wird ein blauer Hintergrundkasten
     in der <MaxChars> entsprechenden Länge angezeigt.

     Wird in <Deposid> ein String übergeben so wird dieser als Vorein-
     stellung angezeigt.

INPUTS
     Deposid  - Der Puffer für den zu lesenden String
     MaxChars - Die maximale Anzahl der zu lesenden Zeichen
     Mode     - Hintergrund ja oder nein

RESULT
     result   - ist NULL wenn das Externe Programm verlassen werden soll.

BUGS

SEE ALSO

             bbs_gets()
             ,
             bbs_getc()
             ,
             bbs_fgetc()
```

1.10 bbs_getc()

```
             BBS.library/bbs_getc

NAME
     bbs_getc -- liest ein Zeichen

SYNOPSIS
     Zeichen bbs_getc()
```

```
     char bbs_getc(void);

FUNCTION
     Wartet bis ein Zeichen eingegeben wurde.

RESULT
     Zeichen - Das eingebene Zeichen oder 0, wenn das Externe Programm
               verlassen werden soll.

BUGS

SEE ALSO

               bbs_gets()
               ,
               bbs_sgets()
               , bbs_fgetc()
```

1.11 bbs_fgetc()

```
               BBS.library/bbs_fgetc

NAME
     bbs_fgetc -- liest ein gefiltertes Zeichen

SYNOPSIS
     Zeichen bbs_fgetc()

     char bbs_fgetc(void);

FUNCTION
     Wartet bis ein darstellbares Zeichen eingegeben wurde.
     Sequencen für Cursorsteuerung werden in ein einzelnes
     Steuerzeichen umgewandelt.

     #define UP        6
     #define DOWN      3
     #define LEFT      4
     #define RIGHT     5
     #define DELETE    7
     #define BACKSPACE 8
     #define RETURN    13

RESULT
     Zeichen - Das eingebene Zeichen oder 0 wenn das Externe Programm
               verlassen werden soll.

BUGS

SEE ALSO

               bbs_gets()
               ,
               bbs_sgets()
```

,
bbs_getc()

1.12 bbs_Wgetc()

```
              BBS.library/bbs_Wgetc

NAME
     bbs_Wgetc -- liest ein Zeichen

SYNOPSIS
     Zeichen bbs_Wgetc(WaitBits)
                       d1

     char bbs_Wgetc(ULONG);

FUNCTION
     Wartet bis ein Zeichen eingegeben wurde.
     Das Warten wird abgebrochen wenn ein Signal auf den
     übergebenen SigBits aufläuft.

RESULT
     Zeichen - Das eingebene Zeichen oder 0 wenn das Externe Programm
               verlassen werden soll.
               Diese Routine gibt eine 2 zurück wenn das Warten durch
               ein Signal abgebrochen wurde.

BUGS

SEE ALSO

              bbs_gets()
              ,
              bbs_sgets()
              ,
              bbs_fgetc()
```

1.13 bbs_Wfgetc()

```
              BBS.library/bbs_Wfgetc

NAME
     bbs_Wfgetc -- liest ein gefiltertes Zeichen

SYNOPSIS
     Zeichen bbs_Wfgetc(WaitBits)
                        d1

     char bbs_Wfgetc(ULONG);

FUNCTION
     Wartet bis ein darstellbares Zeichen eingegeben wurde.
```

```
     Sequencen für Cursorsteuerung werden in ein einzelnes
     Steuerzeichen umgewandelt.

     #define UP        6
     #define DOWN      3
     #define LEFT      4
     #define RIGHT     5
     #define DELETE    7
     #define BACKSPACE 8
     #define RETURN    13

RESULT
     Zeichen - Das eingebene Zeichen oder 0 wenn das Externe Programm
               verlassen werden soll.
               Bei Rückgabe einer 2 wurde das Warten durch ein
               Signal abgebrochen.

BUGS

SEE ALSO

             bbs_gets()
             ,
             bbs_sgets()
             ,
             bbs_getc()
```

1.14 bbs_lookc()

```
             BBS.library/bbs_lookc

NAME
     bbs_lookc -- lesen eines Zeichens (asynchron)

SYNOPSIS

     z = bbs_lookc()

     UBYTE bbs_lookc (void);

FUNCTION

     Diese Funktion bietet die Möglichkeit abzufragen, ob eine Taste
     gedrückt wurde. Im Gegensatz zu z.b. bbs_getc() wird dazu jedoch
     nicht der Programmablauf angehalten, bis eine Taste gedrückt wurde.

     Meldet bbs_looc(), daß eine Taste gedrückt wurde, kann der ASCII-
     Wert des entsprechenden Zeichens mit den beschriebenen Funktionen
     zur Zeichen-Eingabe gelesen werden (z.B. bbs_getc() ).

     Der Rückgabe ist KEIN ASCII-Wert!

RESULT
     z - Wenn derRückgabewert größer als 2 ist, wurde eine Taste
         gedrückt.
```

```
BUGS

SEE ALSO

             bbs_getc()
```

1.15 bbs_graphic()

```
             BBS.library/bbs_graphic

NAME
     bbs_graphic -- zeigt eine ANSI-Datei an

SYNOPSIS
     bbs_graphic(DateiName)
                 a1

     void bbs_graphic(char *);

FUNCTION
     bbs_graphic() gibt eine ANSI-Datei auf dem Bildschirm aus. Dabei
     sollte darauf geachtet werden, daß die Farbcodes in der Form
     "ESC[4X;3Xm" o.ä. gespeichert sind und am Zeilenende jeweils ein
     CR+LF steht.

INPUTS
     DateiName - Name der Datei als C-String

SEE ALSO

             bbs_text()
```

1.16 bbs_text()

```
             BBS.library/bbs_text

NAME
     bbs_text -- zeigt eine Textdatei an

SYNOPSIS
     bbs_text(DateiName)
              a1

     void bbs_text(char *);

FUNCTION
     bbs_text() gibt ein Textfile auf dem Bildschirm aus. Die Textausgabe
     wird angehalten, wenn die vom User eingestellte Zeilenanzahl
     erreicht ist.

INPUTS
```

```
        DateiName - Name des Text-Files als C-String

   SEE ALSO

                bbs_graphic()
```

1.17 bbs_menu()

```
BBS.library/bbs_menu

   NAME
        bbs_menu -- baut ein horizontales Cursor-Shortcut-Menü auf

   SYNOPSIS
        MenuID bbs_menu(FirstMenuItem)
                              a1

        LONG bbs_menu(struct BBSMenu *);

   FUNCTION
        Es wird ein der Menü-Struktur entsprechendes horizontales Menü
        aufgebaut, das sich mit Cursortasten und Shortcuts steuern läßt.

        struct BBSMenu
        {
          struct BBSMenu *Next;
          char *Name;
          LONG MenuID;
          ULONG Private1;
          UBYTE Private2;
        };

        Der ShortCut eines MenuItems wird durch einen '_' im Namen
        ausgewählt, und im Menü farblich abgehoben.
        MenuID ist der Rückgabewert beim Auswählen des entsprechenden
        MenuItems, dieser sollt niemals auf 0 gesetzt werden, da
        der Wert schon für das Verlassen des externen Programms reserviert
        ist.

   INPUTS
        FirstMenuItem - Anfang einer verketteten Menüliste

   RESULT
        MenuID        - Die MenuID des angewählten Items oder 0
                        wenn das externe Programm verlassen werden
                        soll.

   SEE ALSO
```

1.18 LoadUserData()

```
              BBS.library/bbs_LoadUserData

NAME
     bbs_LoadUserData() -- Auslesen von User-Daten

SYNOPSIS

     UD = bbs_LoadUserData(Name)
                           a1

     struct UserDatenExtern *bbs_LoadUserData (char *);

FUNCTION
     Auslesen der User-Daten eines bestimmten Users.
     Die Struktur "UserDatenExtern" wird mit allen wichtigen Daten des
     angegebenen Users gefüllt. Diese können auch verändert und mit
     bbs_SaveUserData() wieder gespeichert werden.

INPUTS
     Name - User-Name

RESULT
     UD   - die geforderten User-Daten

SEE ALSO

              bbs_SaveUserData()
              ,
              bbs_FreeUserData
              ,
              bbs_FirstUser
              ,

              libraries/BBS_library.h
```

1.19 SaveUserData()

```
              BBS.library/bbs_SaveUserData

NAME
     bbs_SaveUserData() -- Speichern von User-Daten

SYNOPSIS

     bbs_SaveUserData( UD )
                       a1

     bbs_SaveUserData ( struct UserDatenExtern * );

FUNCTION
     Abspeichern der mittels bbs_LoadUserData() gelesenen und evtl.
     veränderten User-Daten.

INPUTS
```

```
     UD  - zu speichernde UserDaten

BUGS
     Die Funktion funktioniert nicht, wenn der betroffene User gerade
     online in der Mailbox ist.

SEE ALSO

             bbs_LoadUserData()
             ,
             bbs_FreeUserData()
             ,

             libraries/BBS_library.h
```

1.20 FreeUserData()

```
             BBS.library/bbs_FreeUserData

NAME
     bbs_FreeUserData() -- User-Daten freigeben

SYNOPSIS

     bbs_FreeUserData( UD )
                       a1

     bbs_FreeUserData ( struct UserDatenExtern * );

FUNCTION
     Die Funktion gibt einen mit bbs_LoadUserData gelesenen Datensatz
     wieder frei. Sie muß aufgerufen werden, sobald die User-Daten nicht
     mehr benötigt werden, damit der belegte Speicher der UserDatenExtern-
     Struktur wieder zurückgegeben wird.

INPUTS
     UD  - die von bbs_LoadUserData() erhaltenen User-Daten

SEE ALSO

             bbs_LoadUserData()
             ,
             bbs_SaveUserData()
```

1.21 FirstUser()

```
             BBS.library/bbs_FirstUser

NAME
     bbs_FirstUser -- User-Daten des 1. Users auslesen

SYNOPSIS
```

```
     UserDaten = bbs_FirstUser()

     struct UserDatenExtern *bbs_FirstUser(void);

FUNCTION
     Diese Funktion im Zusammenhang mit bbs_NextUser() dient zum
     schrittweisen durchsehen der User-Liste.

RESULT
     UserDaten - Zeiger auf eine UserDatenExtern-Struktur mit den User-
     Daten des ersten in der Box eingetragenen Users.

SEE ALSO

             bbs_NextUser()
             ,
             bbs_LoadUserData()
```

1.22 NextUser()

```
             BBS.library/bbs_NextUser

NAME
     bbs_NextUser -- Weitere User-Daten lesen

SYNOPSIS

     UserDaten = bbs_NextUser(PrevUser)
                              a1

     struct UserDatenExtern *bbs_NextUser(struct UserDatenExtern *);

FUNCTION
     Lese weitere User-Daten aus der User-Liste.

INPUTS
     PrevUser  - Zeiger auf eine UserDatenExtern-Struktur mit den
                 User-Daten des vorigen Users

RESULT
     UserDaten - Zeiger auf eine UserDatenExtern-Struktur mit den
                 abgeforderten User-Daten.

SEE ALSO

             bbs_FirstUser()
             ,
             bbs_LoadUserData()
```

1.23 MailToUser()

```
             BBS.library/bbs_MailToUser

NAME
     bbs_MailToUser -- Nachricht an einen User schreiben

SYNOPSIS

     bbs_MailToUser(User,Abs,Bet,TextFile)
                    a1   d1  d2   a2

     void bbs_MailToUser(char *, char *, char *, char *);

FUNCTION
     Verschicke eine Textdatei als Nachricht an einen User der Mailbox.

INPUTS
     User     - Username des Empfängers
     Abs      - Absender
     Bet      - Betreff bzw. Überschrift der Nachricht
     TextFile - zu versendende Textdatei

SEE ALSO

             bbs_MailToBrett()
```

1.24 MailToBrett()

```
             BBS.library/bbs_MailToBrett

NAME
     bbs_MailToBrett -- Nachricht in ein Brett schreiben

SYNOPSIS

     bbs_MailToBrett(Brett,Abs,Bet,TextFile)   ???
                     a1    d1  d2    a2

     void bbs_MailToBrett(char *, char *, char *, char *);

FUNCTION
     Schreibt eine Textdatei als Nachricht in das angegebene Brett.

INPUTS
     Brett    - Brettname
     Abs      - Absender der Nachricht
     Bet      - Betreff bzw. Überschrift
     TextFile - File-Name der zu versendenden Textdatei

SEE ALSO

             bbs_MailToUser()
```

1.25 ObtainName()

```
              BBS.library/bbs_ObtainName

NAME
     bbs_ObtainName -- belegen eines Namens

SYNOPSIS

     bbs_ObtainName(String)
                    a1

     bbs_ObtainName(char *);

FUNCTION
     Diese Funktion meldet die alleinige Verwendung eines Strings innerhalb
     der AmBoS-Umgebung für ein externes Programm an.

     Sinnvoll ist dies immer dann, wenn externe Programme Daten-Files
     abspeichern oder lesen wollen. Mit ObtainName() kann dann z.B.
     gesichert werden, daß nicht ein Programm von einem anderen Port aus
     auf das selbe File zugreift. Es kann so gewartet werden, bis der
     gewünschte Zugriff möglich ist, ohne DOS-Fehlermeldungen zu erzeugen.

     Die Funktion verwaltet lediglich die einzelnen Strings, d.h. es wird
     in keiner Weise überprüft um was für einen String (z.B. Filename) es
     sich handelt.

INPUTS
     String - der zu belegende Name

SEE ALSO

              bbs_ReleaseName()
```

1.26 ReleaseName()

```
              BBS.library/bbs_ReleaseName

NAME
     bbs_ReleaseName -- Freigeben eines Namens

SYNOPSIS

     bbs_ReleaseName(String)
                     a1

     bbs_ReleaseName(char *);

FUNCTION
     Gibt den mittels bbs_ObtainName() belegten Namen wieder frei. Das
     externe Programm verliert damit das alleinige "Zugriffsrecht" auf
     diesen String.
```

```
INPUTS
     String - der bereits belegte Name

SEE ALSO

             bbs_ObtainName()
```

1.27 GetBrettType()

```
             BBS.library/bbs_GetBrettType

NAME
     bbs_GetBrettType -- Bret-Typ feststellen

SYNOPSIS

     Typ = bbs_GetBrettType();

     ULONG bbs_GetBrettType (void);

FUNCTION
     Ermittelt den Typ des aktuellen Brettes.

RESULT
     Typ - Nummer eines der in libraries/BBS.h festgelegten Bretttypen.

SEE ALSO

             bbs_FirstBrettInhalt()
             ,
             bbs_BrettInhaltByNumber()
```

1.28 FirstBrettInhalt()

```
             BBS.library/bbs_FirstBrettInhalt

NAME
     bbs_FirstBrettInhalt -- Brettdaten auslesen

SYNOPSIS

     File = bbs_FirstBrettInhalt()

     struct FileExtern *bbs_FirstBrettInhalt(void);

FUNCTION
     Liest den ersten Eintrag des aktuellen Brettes.

RESULT
     File - Zeiger auf eine FileExtern-Struktur mit den Daten des
            ersten Bretteintrages.
```

```
SEE ALSO

             bbs_NextBrettInhalt()
             ,
             bbs_SaveBrettInhalt()
             ,
             bbs_FreeBrettInhalt()
             ,

             bbs_BrettInhaltByNumber()
```

1.29 NextBrettInhalt()

```
             BBS.library/bbs_NextBrettInhalt

NAME
     bbs_NextBrettInhalt -- lese Bretteintrag

SYNOPSIS

     File = bbs_NextBrettInhalt(Prev)
                                a1

     struct FileExtern *bbs_NextBrettInhalt(struct FileExtern *);

FUNCTION
     Liest den nächten Eintrag des aktuellen Brettes.

INPUTS
     Prev - Zeiger auf den vorigen Bretteintrag

RESULT
     File - Zeiger auf den gelesenen Bretteintrag

SEE ALSO

             bbs_FirstBrettInhalt()
             ,
             bbs_FreeBrettInhalt()
```

1.30 SaveBrettInhalt()

```
             BBS.library/bbs_SaveBrettInhalt

NAME
     bbs_SaveBrettInhalt -- schreibe Bretteintrag zurück

SYNOPSIS

     bbs_SaveBrettInhalt(File)
                         a1
```

```
     void bbs_SaveBrettInhalt(struct FileExtern *);

FUNCTION
     Schreibt einen manipulierten Bretteintrag zurück.

INPUTS
     File - Zieger auf den zu schreibenden Datensatz

SEE ALSO

             bbs_FreeBrettInhalt()
```

1.31 FreeBrettInhalt()

```
             BBS.library/bbs_FreeBrettInhalt

NAME
     bbs_FreeBrettInhalt -- Brettinhalt freigeben

SYNOPSIS

     bbs_FreeBrettInhalt(File)
                         a1

     void bbs_FreeBrettInhalt(struct FileExtern *);

FUNCTION
     Gibt einen gelesenen Datensatz wieder frei.

INPUTS
     File - Zeiger auf den freizugebenden Datensatz

SEE ALSO

             bbs_FirstBrettInhalt()
             ,
             bbs_BrettInhaltByNumber()
```

1.32 BrettInhaltByNumber()

```
             BBS.library/bbs_BrettInhaltByNumber

NAME
     bbs_BrettInhaltByNumber -- Lese Bretteintrag

SYNOPSIS

     File = bbs_BrettInhaltByNumber(Nr)
                                    d1

     struct FileExtern *bbs_BrettInhaltByNumber(ULONG);
```

```
   FUNCTION
        Liest einen bestimmten Bretteintrag. Welcher Datensatz gelesen werden
        soll, bestimmt die übergebene Nummer.

   INPUTS
        Nr - Nummer des zu lesenden Datensatzes

   RESULT
        File - Zeiger auf den gelesenen Datensatz

   SEE ALSO

                bbs_FirstBrettInhalt()
                ,
                bbs_NextBrettInhalt()
                ,
                bbs_FreeBrettInhalt()
```

1.33 BBS_library.h

```
/* Eine Externinfo Struktur wird von der Funktion bbs_open() zurückgegeben */

#ifndef EXEC_TYPES_H
#include <exec/types.h>
#endif

#ifndef DOS_DOS_H
#include "dos/dos.h"
#endif

/* Eine Externinfo Struktur wird von der Funktion bbs_open() zurückgegeben */

struct ExternInfo
 {
  BOOL   ConOnly;                       /* Wenn ConLogin ungleich 0 */
  ULONG  StartedFrom;                   /* Hier kann man entnehmen von wo das */
                                        /* Programm gestartet wurde. */
  char  *UserName;                      /* Name des Users */
  char  *City;                          /* Wohnort des Users */
  struct DateStamp LogInTime;           /* Zeitpunkt des Logins */

  ULONG  TotalDownloads;                /* Download Bytes des Users */
  ULONG  TotalUploads;                  /* Upload Bytes des Users */
  ULONG  BaudRate;                      /* BaudRate des Connects */
  ULONG  Lines;                         /* Anzahl der Zeilen des Users */

  ULONG  CallsToday;                    /* Anzahl der Anrufe in der Box Heute */
  ULONG  CallsTotal;                    /* Anzahl der Anrufe in der Box gesammt  ↩
     */
  ULONG  LastCallNr;                    /* Nummer des letzten anrufes des Users  ↩
     */
                                        /* in der Box */
  ULONG  CallNr;                        /* Aktuelle Anrufnummer */

  struct List *TransferListe;           /* Liste der Up- bzw. Downgelodeten */
```

```
                                        /* Files wenn das Programm aus der */
                                        /* Nachupload- oder Nachdownload-Batch   ↩
                                           */
                                        /* gestartet wurde. */
  ULONG  AutoLogOff;                    /* Ist dieser Wert ungleich NULL befindet ↩
      */
                                        /* sich der User in einer Autologoff- */
                                        /* Seuenze, d.h. Eingabeaufforderungen   ↩
                                           */
                                        /* sind tunlichst zu unterlassen... ;-)  ↩
                                           */
  UWORD  Language;                      /* Die Sprache die der User eingestellt  ↩
     */
                                        /* hat. */
  UWORD  DateFormat;                    /* Das bevorzugte Datumsformat des Users ↩
     */
  UWORD  Cosysop;                       /* Wenn ungleich NULL ist der User Sysop ↩
     */
                                        /* oder CoSysop. */
  UWORD  AmBoS_Version;                 /* AmBoS Version */
  UWORD  AmBoS_Revision;                /* AmBoS Revision */
  ULONG  AmBoS_SerialNumber;            /* Seriennummer oder 0 für DEMO-Version  ↩
     */

 };

/* Werte für StartetFrom in der ExternInfo Struktur */

#define FROM_AMENU              0       /* Programm wurd von der GeoNeto- oder   ↩
    */
                                        /* AmBoS-Menu-Oberfläche gestartet. */
#define FROM_NACHLOGIN          1       /* Batchdatei */
#define FROM_GASTLOGIN          2       /* Batchdatei */
#define FROM_VORANTRAG          3       /* Batchdatei */
#define FROM_NACHANTRAG         4       /* Batchdatei */
#define FROM_VORDOWNLOAD        5       /* Batchdatei */
#define FROM_NACHDOWNLOAD       6       /* Batchdatei */
#define FROM_VORUPLOAD          7       /* Batchdatei */
#define FROM_NACHUPLOAD         8       /* Batchdatei */
#define FROM_LOGOFF             9       /* Batchdatei */
#define FROM_RELOGIN            10      /* Batchdatei */

#define FROM_SETUP              11      /* Dieser Modus ist noch nicht implemen ↩
    -*/
                                        /* tiert, er soll dazu verwendet werden  ↩
                                           */
                                        /* ein Kofortables Setup für Externe dem ↩
                                           */
                                        /* Sysop zur verfügunug stellen. */
                                        /* Sollte dieser Modus auftauchen das */
                                        /* Programm am besten sofort beenden. */

/* Werte für Language in der ExternInfo Struktur */

#define LANGUAGE_DEUTSCH        0
#define LANGUAGE_ENGLISH        1
```

```
/* Werte für DateFormat in der ExternInfo Struktur */

#define DATEFORMAT_CDN          0       /* Tag-Monat-Jahr */
#define DATEFORMAT_USA          1       /* Monat-Tag-Jahr */



/* Nodes in der ExternInfo->TransferListe */

struct TransferNode
 {
  struct  TransferNode *tr_Succ;
  struct  TransferNode *tr_Prev;
  UBYTE   tr_Type;                      /* Art der Übertragung Up oder Download  ↩
     */
  BYTE    tr_Pri;
  char   *tr_Name;                      /* Name des File */
  char   *tr_RealName;                  /* Name unter dem das File auf dem
                                           Datenträger zu finden ist */
  char   *tr_BoxPath;                   /* Kompletter Brettpfad */
  char   *tr_DosPath;                   /* Pfad unter dem das File auf dem
                                           Datenträger zu finden ist */
  char   *tr_Uploader;                  /* Der Uploader des Files */
  ULONG   tr_Size;                      /* Länge des Files in Bytes */
  ULONG   tr_CPS;                       /* CPS_Rate bei der Übertragung */
  ULONG   tr_AnzDownloads;              /* Wie oft das File schon downgelodet
                                           wurde */
  ULONG   tr_ProtectedBoard;            /* File liegt in einem durch  ↩
     Zugangsgruppe */
                                        /* geschützten Pfad */
 };

/* Werte für tr_Type in der TransferNode Struktur */

#define TRANSFER_UPLOAD   1
#define TRANSFER_DOWNLOAD 2



struct UserDatenExtern
 {
  APTR  AmBoSPrivat     ;

  char  UserName    [30];
  char  FirstName   [50];
  char  Name        [50];
  char  City       [100];
  char  Street      [60];
  char  PhoneNr     [30];
  char  Fax         [30];
  char  Modem       [30];
  char  Computer    [30];
  char  Substitute  [30];               /* Vertreter */
  char  DLProtocol  [30];
  char  Packer      [30];
```

```
  UWORD BirthYear       ;
  UWORD BirthMonth      ;
  UWORD BirthDay        ;
  ULONG LastLogin       ;               /* In Minuten seit dem 01.01.1978 */
  ULONG NewsDate        ;               /* In Minuten seit dem 01.01.1978 */
  ULONG FirstLogin      ;               /* In Minuten seit dem 01.01.1978 */

  LONG  DLFreeSpace     ;
  ULONG Uploads         ;
  ULONG Downloads       ;

  ULONG LastCall        ;

  UWORD OnlineTime      ;
  UWORD OnlineToday     ;
  UWORD Lines           ;
  UWORD Zone            ;
  UWORD UpDownRatio     ;
  UWORD MaxPMails       ;

  UWORD NumCrashes      ;
  UWORD NumLogins       ;
 };



struct FileExtern
 {
  APTR  AmBoSPrivat;

  ULONG Number    ;                     /* Nummer des Files im Brett */
  UWORD Delete    ;                     /* kann gesetzt werden */
  UWORD Markiert  ;                     /* Eintrag ist markiert */

  UWORD FileType  ;                     /* File oder Mail */
  UWORD BrettType ;                     /* Art des aktuellen Brettes */
  UWORD Downloads ;                     /* Anzahl der Zugriffe auf den Eintrag   ↩
     */
  ULONG UploadDate;                     /* In Minuten seit dem 01.01.1978 */
  ULONG CreateDate;                     /* In Minuten seit dem 01.01.1978 */
  ULONG Size      ;                     /* Länge von Binärfiles */
  ULONG Lines     ;                     /* Anzahl der Zeilen bei Mails */

  char *Uploader  ;                     /* Name des Uploaders */
  char *RealName  ;                     /* Name unter dem das File auf der */
                                        /* Platte zu finden ist. */
  char *BoxPath   ;                     /* Brettpfad */
  char *DosPath   ;                     /* Dospfad des aktuellen Brettes */

  char ReadMeFile[42];                  /* FileName des Readmetextes */
                                        /* kann verändert werden */
  char BoxName[42]   ;                  /* Name unter dem ein File in der Box */
                                        /* angezeigt wird. */
                                        /* kann verändert werden */
  char Comment[8][42];                  /* Beschreibung des Eintrages */
                                        /* kann verändert werden */
```

```
 };

 /* Filetypen */

#define FileType_MESSAGE                1
#define FileType_BIN                    2



struct BrettDatenExtern
 {
  APTR  AmBoSPrivat;

  char  BrettName     [40];
  char  BrettPfad     [256];

  char  SchreibGruppe [40];
  char  ZugangsGruppe [40];
  char  LeseGruppe    [40];
  char  Verwalter     [40];
  char  BrettPasswort [40];

  ULONG LetzterEintrag  ;
  ULONG BrettFlags      ;
  UBYTE BrettTyp        ;
  UBYTE Locked          ;
  UBYTE Area            ;
  UBYTE NoRatio         ;
 };

 /* Bretttypen */

#define BrettType_NoBrett               0
#define BrettType_Asc                   1
#define BrettType_Bin                   2
#define BrettType_AscBin                3
#define BrettType_Head                  4
#define BrettType_Extern                5
#define BrettType_PM                    6

 /* Datenstruktur für bbs_menu() */

struct BBSMenu
 {
   struct BBSMenu *Next;
   char *Name;
   LONG MenuID;         /* Niemals auf 0 setzen !       */

   ULONG Private1;      /* Immer mit 0 initialisieren ! */
   UBYTE Private2;      /* Immer mit 0 initialisieren ! */
 };
```

1.34 BBS.fd

AmBoS FD-File

Anmerkung zu "bbs_printf":

Diese Funktion kann "normalerweise" nicht mit FD-Format beschrieben
werden, da es zu alt ist. Daher kann es zu Problemen kommen, wenn
bbs_printf() in eine andere Programmiersprache übernommen werden soll.
Gegebenenfalls hilft vielleicht ein Blick in die compiler-abhängigen
Inlude-Files zu anderen Systemfunktionen wie z.B. dem printf() aus der
Standardbibliothek.

FD-File:

```
##base _BBSBase
##bias 30
##private
bbs_Private1()()
bbs_Private2()()
##public
bbs_open(PortName)(a1)
bbs_close()()
bbs_puts(String)(a1)
bbs_gets(Deposid,MaxChars,Mode)(a1/d1/d2)
bbs_sgets(Deposid,MaxChars,Mode)(a1/d1/d2)
##private
bbs_Private3()()
##public
bbs_getc()()
bbs_fgetc()()
bbs_menu(MenuDaten)(a1)
bbs_graphic(FileName)(a1)
bbs_text(FileName)(a1)
##private
bbs_Private4()()
##public
bbs_rputs(String)(a1)
##private
bbs_Private5()()
bbs_Private6()()
bbs_Private7()()
##public
bbs_Wgetc(WaitBits)(d1)
bbs_Wfgetc(WaitBits)(d1)
##private
bbs_Private8()()
bbs_Private9()()
bbs_Private10()()
##public
bbs_printf(String, Tags)(a1/a2)
bbs_Wgets(Deposid,MaxChars,Mode,Bits)(a1/d1/d2/d3)
##private
bbs_Private11()()
##public
bbs_lookc()()
bbs_FirstUser()()
```

```
bbs_NextUser(User)(a1)
bbs_ObtainName(Name)(a1)
bbs_ReleaseName(Name)(a1)
bbs_LoadUserData(UserName)(a1)
bbs_SaveUserData(UserData)(a1)
bbs_FreeUserData(UserData)(a1)
bbs_MailToUser(UserName, Absender,Betreff, TextFile)(a1/d1/d2/a2)
bbs_MailToBrett(Brett,Absender,Betreff, TextFile)(a1/d1/d2/a2)
bbs_GetBrettType()()
bbs_FirstBrettInhalt()()
bbs_NextBrettInhalt(BrettInhalt)(a1)
bbs_BrettInhaltByNumber(Number)(d1)
bbs_FreeBrettInhalt(BrettInhalt)(a1)
bbs_SaveBrettInhalt(BrettInhalt)(a1)
##end
```

1.35 BBS_pragmas.h

```
SAS-C Pragmas zur BBS.library
-----------------------------

(Filename: "include:pragmas/BBS_pragmas.h")

#pragma libcall BBSBase bbs_setdata 1E 901
#pragma libcall BBSBase bbs_readdata 24 901
#pragma libcall BBSBase bbs_open 2A 901
#pragma libcall BBSBase bbs_close 30 0
#pragma libcall BBSBase bbs_puts 36 901
#pragma libcall BBSBase bbs_gets 3C 21903
#pragma libcall BBSBase bbs_sgets 42 21903
#pragma libcall BBSBase bbs_clear 48 0
#pragma libcall BBSBase bbs_getc 4E 0
#pragma libcall BBSBase bbs_fgetc 54 0
#pragma libcall BBSBase bbs_menu 5A 901
#pragma libcall BBSBase bbs_graphic 60 901
#pragma libcall BBSBase bbs_text 66 901
#pragma libcall BBSBase bbs_getquotetext 6C 0
#pragma libcall BBSBase bbs_getsigtext 72 0
#pragma libcall BBSBase bbs_savetext 78 901
#pragma libcall BBSBase bbs_cputs 7E 901
#pragma libcall BBSBase bbs_hgets 84 3A21905
#pragma libcall BBSBase bbs_Wgetc 8A 101
#pragma libcall BBSBase bbs_Wfgetc 90 101
#pragma libcall BBSBase bbs_Afgetc 96 0
#pragma libcall BBSBase bbs_SaveColor 9C 0
#pragma libcall BBSBase bbs_PutColor A2 0
#pragma tagcall BBSBase bbs_printf A8 A902
#pragma libcall BBSBase bbs_Wgets AE 321904
#pragma libcall BBSBase bbs_lookc BA 0
#pragma libcall BBSBase bbs_rputs 72 901
#pragma libcall BBSBase bbs_FirstUser C0 00
#pragma libcall BBSBase bbs_NextUser C6 901
#pragma libcall BBSBase bbs_ObtainName CC 901
#pragma libcall BBSBase bbs_ReleaseName D2 901
#pragma libcall BBSBase bbs_LoadUserData D8 901
```

```
#pragma libcall BBSBase bbs_SaveUserData DE 901
#pragma libcall BBSBase bbs_FreeUserData E4 901
#pragma libcall BBSBase bbs_MailToUser EA A21904
#pragma libcall BBSBase bbs_MailToSysInfo F0 A2103
#pragma libcall BBSBase bbs_GetBrettType F6 00
#pragma libcall BBSBase bbs_FirstBrettInhalt FC 00
#pragma libcall BBSBase bbs_NextBrettInhalt 102 901
#pragma libcall BBSBase bbs_BrettInhaltByNumber 108 101
#pragma libcall BBSBase bbs_FreeBrettInhalt 10E 901
#pragma libcall BBSBase bbs_SaveBrettInhalt 114 901
```

1.36 BBS_protos.h

```
C-Prototypen zur BBS.library
----------------------------

/* BBS.library C-prototrypes */

struct ExternInfo *bbs_open (char *);
void bbs_close (void);

int bbs_printf (char *, ...);
void bbs_puts (char *);
int bbs_gets (char *, LONG, LONG);
int bbs_sgets (char *, LONG, LONG);
char bbs_getc (void);
char bbs_fgetc (void);
char bbs_Wfgetc (ULONG);
UBYTE bbs_lookc (void);

struct UserDatenExtern *bbs_LoadUserData(char *);
void bbs_SaveUserData(struct UserDatenExtern *);
void bbs_FreeUserData(struct UserDatenExtern *);
struct UserDatenExtern *bbs_FirstUser(void);
struct UserDatenExtern *bbs_NextUser(struct UserDatenExtern *);

void bbs_MailToUser(char *, char *, char *,char *);
void bbs_MailToBrett(char *, char *, char *);
ULONG bbs_GetBrettType(void);
struct FileExtern *bbs_FirstBrettInhalt(void);
struct FileExtern *bbs_NextBrettInhalt(struct FileExtern *);
void bbs_SaveBrettInhalt(struct FileExtern *);
void bbs_FreeBrettInhalt(struct FileExtern *);
struct FileExtern *bbs_BrettInhaltByNumber(ULONG);

void bbs_graphic (char *);
void bbs_text (char *);
LONG bbs_menu (struct BBSMenu *);

void bbs_ObtainName(char *);
void bbs_ReleaseName(char *);
```