SuperPlay_Ref_GER

COLLABORATORS

	TITLE : SuperPlay_Ref_GER	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 28, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 SuperPlay_Ref_GER **1**

1.1 SuperPlay_Ref_GER.doc . . . 1
1.2 superplay.library/--background-- . . . 1
1.3 superplay.library/SPL_AllocHandle . . . 2
1.4 superplay.library/SPL_FreeHandle . . . 3
1.5 superplay.library/SPL_StopReplay . . . 3
1.6 superplay.library/SPL_FreeResources . . . 4
1.7 superplay.library/SPL_SuperPlay . . . 4
1.8 superplay.library/SPL_SuperWrite . . . 5
1.9 superplay.library/SPL_InitHandleAsDOS . . . 6
1.10 superplay.library/SPL_InitHandleAsClip . . . 7
1.11 superplay.library/SPL_SetWriteType . . . 7
1.12 superplay.library/SPL_GetErrorString . . . 8
1.13 superplay.library/SPL_SetWriteName . . . 8
1.14 superplay.library/SPL_FileInfoRequest . . . 9
1.15 superplay.library/SPL_SetReqIOWindow . . . 10
1.16 superplay.library/SPL_ReadPlayData . . . 10
1.17 superplay.library/SPL_ContinueReplay . . . 11
1.18 superplay.library/SPL_FastForward . . . 11
1.19 superplay.library/SPL_FastBackward . . . 12
1.20 superplay.library/SPL_GetSampleList . . . 12
1.21 superplay.library/SPL_SetSampleList . . . 13
1.22 superplay.library/SPL_GetFileType . . . 14

Chapter 1

SuperPlay_Ref_GER

1.1 SuperPlay_Ref_GER.doc

```
--background--
SPL_AllocHandle()
SPL_FreeHandle()
SPL_StopReplay()
SPL_FreeResources()
SPL_SuperPlay()
SPL_SuperWrite()
SPL_InitHandleAsDOS()
SPL_InitHandleAsClip()
SPL_SetWriteType()
SPL_GetErrorString()
SPL_SetWriteName()
SPL_FileInfoRequest()
SPL_SetReqIOWindow()
SPL_ReadPlayData()
SPL_ContinueReplay()
SPL_FastForward()
SPL_FastBackward()
SPL_GetSampleList()
SPL_SetSampleLisSet()
SPL_GetFileType()
```

1.2 superplay.library/--background--

```
VERSION
     $VER: SuperPlay_Ref_GER.doc V6.3 (3.4.97)

COPYRIGHT
     © 1995-97 by Andreas R. Kleinert. All rights reserved.

     - Feel free to translate this Doc-File into other languages. -

GENERAL
           Andreas R. Kleinert,
           Sandstrasse 1,
```

D-57072 Siegen,
Germany.

EMail: Fido Andreas Kleinert 2:2457/350.18
Usenet/InterNet Andreas_Kleinert@superview.ftn.neckar-alb.de

If nothing else works, try one of these Fido-InterNet gateways:

Andreas_Kleinert@p10.f435.n2457.z2.fido.sub.org (in Germany)
Andreas_Kleinert@p10.f435.n2457.z2.fidonet.org (USA or other)

```
HISTORY
     V2     bug-fixed ClipBoard-Support
            with the supplied SPObjects (ak)
     V3     created working combination of
            SPL_SetSampleList() and SuperWrite()
            calls (ak)
     V4     filetype recognition without loading (ak)
     V4.6   Autodoc format
            SetWriteName(): Correct type of write_name
            Change two SVL_ prefix to SPL_
            Add version behind NAME            (indy)
     V5.1   SetWriteName() parameter must be UBYTE * (ak)
     V5.2   add "()" to function names in text for better autodocs
            support(indy)
     V6.3   bumped version
            updated copyright note
            updated email address list (ak)
```

1.3 superplay.library/SPL_AllocHandle

```
NAME
     SPL_AllocHandle -- Handle reservieren  (V1)

SYNOPSIS

     APTR SPL_AllocHandle(APTR future)
     D0   -$1e            A1

FUNCTION

     Reserviert einen Handle zum Zugriff auf ein Sample/Module über
     SPObjects.

INPUT(S)

     future - derzeit immer NULL

RESULT

     Ein Zeiger auf einen neu reservierten Handle oder NULL, falls
     die Reservierung fehlschlägt.

WARNING
```

```
     Überprüfe, ob das Ergebnis NULL oder ungleich NULL ist!

SEE ALSO

     SPL_FreeResources(), SPL_FreeHandle()
```

1.4 superplay.library/SPL_FreeHandle

```
NAME
     SPL_FreeHandle -- Handle freigeben (V1)

SYNOPSIS

     VOID SPL_FreeHandle(APTR handle)
     D0   -$24           A1

FUNCTION

    Beendet das Abspielen und gibt alle Ressourcen sowie den zuvor mit
    SPL_AllocHandle() reservierten Handle frei.

INPUT(S)

     handle - ein gültiger Handle

RESULT

     -

SEE ALSO

     SPL_AllocHandle(), SPL_StopReplay(), SPL_FreeResources()
```

1.5 superplay.library/SPL_StopReplay

```
NAME
     SPL_StopReplay -- Beendet Abspielen des Samples/Modules  (V1)

SYNOPSIS

     VOID SPL_StopReplay(APTR handle)
     D0   -$2a           A1

FUNCTION

     Beendet das Abspielen des mit den Handle angegebenen Sample/Module.
     Einige SPObjects ermöglichen es das Sample/Module durch Aufruf
     von SPL_ContinueReplay() weiter abspielen zu lassen, nachdem es
     mit SPL_StopReplay() gestoppt worden ist.

INPUT(S)
```

```
     handle - ein gültiger Handle

RESULT

     -

SEE ALSO

     SPL_ContinueReplay(), SPL_FreeResources(), SPL_FreeHandle()
```

1.6 superplay.library/SPL_FreeResources

```
NAME
     SPL_FreeResources -- Freigabe nicht benötigter Ressourcen  (V1)

SYNOPSIS

     VOID SPL_FreeResources(APTR handle)
     D0   -$30              A1

FUNCTION

     Gibt alle Ressourcen - des mit den Handle angegebenen Samples/Modules
     frei - die nicht benötigt werden, um es einfach nur abzuspielen.
     Das Abspielen des Samples/Modules wird weder gestoppt noch
     unterbrochen.

INPUT(S)

     handle - ein gültiger Handle

RESULT

     -

SEE ALSO

     SPL_AllocHandle(), SPL_StopReplay(), SPL_FreeHandle()
```

1.7 superplay.library/SPL_SuperPlay

```
NAME
     SPL_SuperPlay -- Lädt und spielt Sample/Module (V1)

SYNOPSIS

     ULONG SPL_SuperPlay(APTR handle, char *filename)
     D0    -$36          A1           A2

FUNCTION
```

```
     Lädt und spielt das mit den Dateinamen spezifizierte Sample/Module
     ab.
     Das Handle wird initialisiert und das passende SPObject geöffnet
     und zum Abspielen des Sample/Module verwendet.

     Das Abspielen kann entweder durch vollständige Freigabe des
     Handles(SPL_FreeHandle()) beendet oder mit SPL_StopReplay()
     unterbrochen werden.

INPUT(S)

     handle   - ein gültiger Handle
     filename - ein gültiger AmigaDOS Dateipfad und -name

RESULT

     NULL oder eine SPERR-Fehlernummer.

SEE ALSO

     SPL_AllocHandle(), SPL_StopReplay(), SPL_ContinueReplay(),
     SPL_FastForward(), SPL_FastBackward(), SPL_FreeHandle()
```

1.8 superplay.library/SPL_SuperWrite

```
NAME
     SPL_SuperWrite -- Sample/Module speichern (V1)

SYNOPSIS

     ULONG SPL_SuperWrite(APTR handle, APTR source_handle)
     D0    -$3c           A1           A2

FUNCTION

     Normalerweise wird ein Sample/Module mit SPL_SuperPlay() geladen
     und abgespielt: Abgesehen von SPL_AllocHandle()/SP_FreeHandle() sind
     hierzu sind keine weiteren Funktionsaufrufe zum Lesen und
     Abpielen der Daten nötig.

     Zum Schreiben - also der Konvertierung eines Samples/Modules -
     müssen folgende Funktionsaufrufe ausgeführt werden:

     source_handle = SPL_AllocHandle(N);
     result        = SPL_ReadPlayData(source_handle, source_name);
     dest_handle   = SPL_AllocHandle(N);
     /* result     = SPL_InitHandleAsDOS(dest_handle, N); */ /* default */
     result        = SPL_SetWriteName(dest_handle, dest_name, N);
     result        = SPL_SetWriteType(dest_handle, dest_type, N);
     result        = SPL_SuperWrite(dest_handle, source_handle);
     SPL_FreeHandle(dest_handle);
     SPL_FreeHandle(source_handle);


     Wichtig: Überprüfe den "result" Wert NACH JEDEM Funktionsaufruf
```

```
               (siehe Beispiel Quellcodes) !

     Alle möglichen Werte für dest_type können in der SPOBject-List
     der SuperPlayBase gefunden werden.
     Diese Werte VERÄNDERN sich bei jeder Neuinitialisierung der
     superplay.library, deshalb muß man sie bei jeden Öffnen der
     Library erneut auslesen.

     WENN "source_handle" = NULL,
     wird überprüft, ob eine SampeList mittels "SPL_SetSampleList()"
     gesetzt worden ist wenn dies der Fall ist wird diese
     SampleList komplett abgespeichert.

     Ein wird empfohlen den alten Weg zur Konvertierung zwischen
     Dateiformaten und den neuen Weg zum Abspeichern einzelner(selbst
     erstellter) SampleLists zu verwenden.

INPUT(S)

     handle        - ein gültiger Handle          (zum Schreiben)
     source_handle - ein anderer gültiger Handle  (zum Lesen)

RESULT

     NULL oder eine SPERR-Fehlernummer.

SEE ALSO

     SPL_AllocHandle(), SPL_ReadPlayData(), SPL_FreeHandle()
```

1.9 superplay.library/SPL_InitHandleAsDOS

```
NAME
     SPL_InitHandleAsDOS -- Initialisiert Handle für AmigaDOS  (V1)

SYNOPSIS

     ULONG SPL_InitHandleAsDOS(APTR handle, APTR future)
     D0    -$42                A1           A2

FUNCTION

     Initialisiert den Handle für den Zugriff über AmigaDOS hierbei
     wird der angegebene Dateiname benutzt.

     Eine andere Möglichkeit besteht darin den Handle für den
     Clipboard Zugriff zu initialisieren(abhängig vom jeweiligen
     SPObject, z.B. IFF-8SVX).

INPUT(S)

     handle - ein gültiger Handle
     future - derzeit immer NULL

RESULT
```

```
     NULL oder eine SPERR-Fehlernummer.

SEE ALSO

     SPL_InitHandleAsClip()
```

1.10 superplay.library/SPL_InitHandleAsClip

```
NAME
     SPL_InitHandleAsClip -- Initialisiert Handle für ClipBoard  (V1)

SYNOPSIS

     ULONG SPL_InitHandleAsClip(APTR handle, APTR future)
     D0    -$48                 A1           A2

FUNCTION

     Initialisiert einen Handle für den Zugriff aufs Clipboard, dabei
     wird ein (möglicherweise) angegebener AmigaDOS Datename ignoriert.

     Die beinahe fast immer benutze Möglichkeit besteht in der
     Initialisierung des Handle zum AmigaDOS Zugriff(wird von
     allen SPObjects unterstützt).

INPUT(S)

     handle - ein gültiger Handle
     future - derzeit immer NULL

RESULT

     NULL oder ein SPERR-Fehlercode.

SEE ALSO

     SPL_InitHandleAsDOS()
```

1.11 superplay.library/SPL_SetWriteType

```
NAME
     SPL_SetWriteType -- Schreibtyp setzen  (V1)

SYNOPSIS

     ULONG SPL_SetWriteType(APTR handle, ULONG write_type, APTR future)
     D0    -$4e             A1           A2                A3

FUNCTION

     Setzt den Schreibtyp für einen SPL_SuperWrite() Aufruf.
```

```
     Siehe dortige Beschreibung und Beispiel Quellcodes für weitere
     und detailliertere Informationen.

INPUT(S)

     handle     - ein gültiger Handle
     write_type - ein gültiger - zeitlich begrenzter - write_type Code
                  aus der SPObject List
     future     - derzeit immer NULL

RESULT

     NULL oder eine SPERR-Fehlernummer.

SEE ALSO

     SPL_SuperWrite()
```

1.12 superplay.library/SPL_GetErrorString

```
NAME
     SPL_GetErrorString -- Liefert Fehlertext  (V1)

SYNOPSIS

     char * SPL_GetErrorString(ULONG error_code)
     D0    -$54                A1

FUNCTION

     Liefert den Fehlertext für eine bestimmte Fehlernummer die
     von einer der superplay.library Funktionen zurückgegeben worden
     ist.

INPUT(S)

     error_code - eine SPERR-Fehlernummer

RESULT

     Nur-Lese Zeiger auf einen SPERR-Fehlertext

SEE ALSO

     -
```

1.13 superplay.library/SPL_SetWriteName

```
NAME
     SPL_SetWriteName -- Dateinamen für SPL_SuperWrite() setzen  (V1)

SYNOPSIS
```

```
     ULONG SPL_SetWriteNameClip(APTR handle, UBYTE *write_name, APTR future)
     D0    -$5a                 A1           A2                 A3

FUNCTION

     Setzt den Dateinamen für einen SPL_SuperWrite() Aufruf.
     Siehe dortige Beschreibung und Beispiel Quellcodes für weitere
     und detailliertere Informationen.

INPUT(S)

     handle     - ein gültiger Handle
     write_name - ein gültiger AmigaDOS Dateipfad und -name
     future     - derzeit immer NULL


RESULT

     NULL oder eine SPERR-Fehlernummer.

SEE ALSO

     SPL_SuperWrite()
```

1.14 superplay.library/SPL_FileInfoRequest

```
NAME
     SPL_FileInfoRequest -- Zeigt Informationen zum Sample/Module  (V1)

SYNOPSIS

     ULONG SPL_FileInfoRequest(APTR handle, struct Window *window,
     D0    -$60                A1           A2

                               APTR future)
                               A3

FUNCTION

     Gibt ein Infofenster mit mehr oder weniger detaillierten Informationen
     über das derzeit geladene Sample/Module aus.
     Es ist möglich einen Zeiger auf ein Fenster anzugeben mit dem
     spezifiziert wird wo dieses Infofenster erscheinen soll.

INPUT(S)

     handle - ein gültiger Handle
     window - ein gültiger Fensterzeiger oder NULL
     future - derzeit immer NULL


RESULT

     NULL oder eine SPERR-Fehlernummer.
```

```
SEE ALSO

     SPL_SetReqIOWindow()
```

1.15 superplay.library/SPL_SetReqIOWindow

```
NAME
     SPL_SetReqIOWindow -- Default Fenster für Dialogfenster setzen  (V1)

SYNOPSIS

     ULONG SPL_SetReqIOWindow(APTR handle, struct Window *window)
     D0    -$66               A1           A2

FUNCTION

     Setzt ein Default-Fenster (default: IntuitionBase->FirstWindow)
     für alle Dialogfenster.

INPUT(S)

     handle - ein gültiger Handle
     window - ein gültiger Fensterzeiger oder NULL

RESULT

     NULL oder eine SPERR-Fehlernummer.

SEE ALSO

     SPL_FileInfoReq
```

1.16 superplay.library/SPL_ReadPlayData

```
NAME
     SPL_ReadPlayData -- Lädt Daten eines Samples/Modules  (V1)

SYNOPSIS

     ULONG SPL_ReadPlayData(APTR handle, char *filename)
     D0    -$6c             A1           A2

FUNCTION

     Lädt die Daten eines mittels des Dateinamen spezifizierten
     Samples/Modules aber spielt diese NICHT ab.
     Das Handle wird initialisiert und das entsprechende SPObject
     geöffnet, um mit SPL_SuperWrite() auf die Sample/Module Daten
     zugreifen zu können.

INPUT(S)
```

```
     handle   - ein gültiger Handle
     filename - ein gültiger AmigaDOS Dateipfad und -name

RESULT

     NULL oder eine SPERR-Fehlernummer.

SEE ALSO

     SPL_SetWriteName(), SPL_SetWriteType(), SPL_SuperWrite()
```

1.17 superplay.library/SPL_ContinueReplay

```
NAME
     SPL_ContinueReplay -- Abspielen fortsetzen  (V1)

SYNOPSIS

     ULONG SPL_ContinueReplay(APTR handle)
     D0    -$72               A1

FUNCTION

     Einige SPObjects bieten die Möglichkeit das durch SPL_StopReplay()
     unterbrochene Abspielen des Samples/Modules durch Aufruf
     dieser Funktion fortzuführen.

INPUT(S)

     handle - ein gültiger Handle

RESULT

     NULL oder eine SPERR-Fehlernummer.

SEE ALSO

     SPL_SuperPlay(), SPL_StopReplay()
```

1.18 superplay.library/SPL_FastForward

```
NAME
     SPL_FastForward -- Schnellvorlauf  (V1)

SYNOPSIS

     ULONG SPL_FastForward(APTR handle)
     D0    -$78            A1

FUNCTION
```

```
     Einige SPObjects unterstützen die auch von Kassettenrecordern
     bekannte Möglichkeit des Schnellvor- bzw. rücklaufs durch
     ein Sample/Module.

INPUT(S)

     handle - ein gültiger Handle

RESULT

     NULL oder eine SPERR-Fehlernummer.

SEE ALSO

     SPL_FastBackward()
```

1.19 superplay.library/SPL_FastBackward

```
NAME
     SPL_FastBackward -- Schnellrücklauf  (V1)

SYNOPSIS

     ULONG SPL_FastBackward(APTR handle)
     D0    -$7e             A1

FUNCTION

     Einige SPObjects unterstützen die auch von Kassettenrecordern
     bekannte Möglichkeit des Schnellvor- bzw. rücklaufs durch
     ein Sample/Module.

INPUT(S)

     handle - ein gültiger Handle

RESULT

     NULL oder eine SPERR-Fehlernummer.

SEE ALSO

     SPL_FastForward()
```

1.20 superplay.library/SPL_GetSampleList

```
NAME
     SPL_GetSampleList -- Zugriff auf SampleList  (V2)

SYNOPSIS

     ULONG SPL_GetSampleList(APTR handle, struct SPO_SampleList **list)
```

```
     D0    -$84              A1           A2

FUNCTION

     Während bzw. nachdem Laden einer Sample-Datei erstellen SPObjects
     der Version 2 eventuell eine spezielle Liste aller Samples die in
     dieser Datei enthalten sind.
     Dies wird in der Regel bei einer Sample-Datei ein Sample sein
     und eine ganze Reihe von Samples bei einen Module.

     Nicht alle SPObjects, speziell nicht alle ModuleType-SPObjects,
     unterstützen dies und werden in diesen Fall eine Fehlernummer oder
     eine leere SampleList als Ergebnis liefern.

INPUT(S)

     handle - ein gültiger Handle
     list   - ein Zeiger auf einen SPO_SampleList Zeiger

RESULT

     NULL oder eine SPERR-Fehlernummer.

SEE ALSO

     SPL_SetSampleList()
```

1.21 superplay.library/SPL_SetSampleList

```
NAME
     SPL_SetSampleList -- SampleList setzen  (V2)

SYNOPSIS

     ULONG SPL_SetSampleList(APTR handle, struct SPO_SampleList *list)
     D0    -$8a              A1           A2

FUNCTION

     Zum Speichern von Sample-Dateien mit SPObjects der Version 2 kann
     eine spezielle Liste aller Samples, die in dieser Datei erscheinen
     sollen, verwendet werden.

     Dies wird in der Regel bei einer Sample-Datei ein Sample sein
     und eine ganze Reihe von Samples bei einen Module.

     Nicht alle SPObjects, speziell nicht alle ModuleType-SPObjects,
     unterstützen dies und könnten stattdessen ein
     SPERR_ACTION_NOT_SUPPORTED als Fehler liefern.

INPUT(S)

     handle - ein gültiger Handle
     list   - ein Zeiger auf eine SPO_SampleList
```

```
RESULT

     NULL oder eine SPERR-Fehlernummer.

SEE ALSO

     SPL_GetSampleList()
```

1.22 superplay.library/SPL_GetFileType

```
NAME
     SPL_GetFileType -- Liefert Dateitypen-Code  (V4)

SYNOPSIS

     ULONG SPL_GetFileType(APTR handle, char *filename, ULONG *filetype)
     D0    -$90            A1           A2              A3

FUNCTION

     Findet den SuperPlay spezifischen Dateitypen-Code
     (der bei jeder (neuen)-Initialisierung der Library neu definiert
     wird) oder SP_FILE_TYPE_UNKNOWN (== NULL == FALSE).

     Benutzen Sie folgende Funktionsaufrufe für eine einfache Überprüfung:

     handle   = SPL_AllocHandle(N);
     SPerr    = SPL_GetFileType(handle, filename, &filetype);
                SPL_FreeHandle(handle);

     Dieser Handle darf NICHT für weitere Operationen mit dieser
     Datei benutzt werden(Datei wird einmal geöffnet und zweimal
     überprüft aber nur einmal geschlossen, usw.)

     Initialisierungs Funktionen wie SPL_InitHandleAsClip() sind
     gar nicht erlaubt.

     Wichtig, diese Funktion setzt nur die FILETYPES und nicht
     die SUBTYPES. Zum Schreiben müssen sie z.B. diese SUBTYPES
     spezifizieren.

     FILETYPES sind nur für eine einfache Idendifikation gedacht
     und geben nur den allgemeinen Typ an(8SVX, ST).

INPUT(S)

     handle   - ein gültiger Handle
     filename - ein gültiger AmigaDOS Dateipfad- und -name
     filetype - Zeiger auf ULONG für SP_FILETYPE-Wert

RESULT

     NULL oder eine SPERR-Fehlernummer.

SEE ALSO
```

SPL_AllocHandle(), SPL_FreeHandle()