

**SuperPlay\_Ref\_GER**

**COLLABORATORS**

	<i>TITLE :</i> SuperPlay_Ref_GER		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		February 12, 2023	

**REVISION HISTORY**

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>SuperPlay_Ref_GER</b>	<b>1</b>
1.1	SuperPlay_Ref_GER.doc . . . . .	1
1.2	superplay.library/--background-- . . . . .	2
1.3	superplay.library/SPL_AllocHandle . . . . .	2
1.4	superplay.library/SPL_FreeHandle . . . . .	3
1.5	superplay.library/SPL_StopReplay . . . . .	4
1.6	superplay.library/SPL_FreeResources . . . . .	4
1.7	superplay.library/SPL_SuperPlay . . . . .	5
1.8	superplay.library/SPL_SuperWrite . . . . .	6
1.9	superplay.library/SPL_InitHandleAsDOS . . . . .	7
1.10	superplay.library/SPL_InitHandleAsClip . . . . .	8
1.11	superplay.library/SPL_SetWriteType . . . . .	9
1.12	superplay.library/SPL_GetErrorString . . . . .	9
1.13	superplay.library/SPL_SetWriteName . . . . .	10
1.14	superplay.library/SPL_FileInfoRequest . . . . .	10
1.15	superplay.library/SPL_SetReqIOWindow . . . . .	11
1.16	superplay.library/SPL_ReadPlayData . . . . .	12
1.17	superplay.library/SPL_ContinueReplay . . . . .	12
1.18	superplay.library/SPL_FastForward . . . . .	13
1.19	superplay.library/SPL_FastBackward . . . . .	14
1.20	superplay.library/SPL_GetSampleList . . . . .	14
1.21	superplay.library/SPL_SetSampleList . . . . .	15
1.22	superplay.library/SPL_GetFileType . . . . .	15

---

## 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.

E-Mail: 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 SPOjects (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 SPOjects.

## 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 SPOjects 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 SPObjekt 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 Abspielen 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 benutzte 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 SPOjects 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 SPOjects 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 SPOjects, speziell nicht alle ModuleType-SPOjects, 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 SPObjets 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 einem Module.

Nicht alle SPObjets, speziell nicht alle ModuleType-SPObjets, 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 Identifikation 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()
```