PlayStation®2 EE Library Reference
Release 2.4.3

Sound Libraries

Summary Table of Contents

About This Manual **v**
Changes Since Last Release v
Related Documentation v
Typographic Conventions vi
Developer Support vi

Chapter 1: CSL MIDI Stream Generation **1-1**
Structures 1-3
Functions 1-4

Chapter 2: CSL SE Stream Generation (for EE) **2-1**
Functions 2-3

Chapter 3: CSL Software Synthesizer **3-1**
Structures 3-3
Functions 3-8
Callback Functions 3-27

Chapter 4: CSL Line-out for EE **4-1**
Structures 4-3
Functions 4-4

Chapter 5: Low-Level Sound Library **5-1**
Functions 5-3

Chapter 6: Standard Kit Library/Sound System **6-1**
Functions 6-3

Chapter 7: SPU2 Local Memory Management Library **7-1**
Functions 7-3

About This Manual

This is the Runtime Library Release 2.4.3 version of the *PlayStation®2 EE Library Reference - Sound Libraries* manual.

The purpose of this manual is to define all available PlayStation®2 EE sound library structures and functions. The companion *PlayStation®2 EE Library Overview - Sound Libraries* describes the structure and purpose of the libraries.

Changes Since Last Release

Chapter 3: CSL Software Synthesizer

- In sceSSyn_ATick(), an error in function name has been corrected.

Incorrect:	sceSSyn_Atick
Correct :	sceSSyn_ATick

- In sceSSyn_ATick(), the description of "Calling Conditions" has been corrected.
  (There is no change in function implementation).
- In the following functions, a description on calling in a multithreaded environment has been added.

  sceSSyn_BreakAtick(), sceSSyn_ClearBreakAtick(), sceSSynInit(), sceSSyn_Load(),

  sceSSyn_PrepareParameter(), sceSSyn_SendRpnMsg(), sceSSyn_SendShortMsg(),

  sceSSyn_SetChPriority(), sceSSyn_SetMasterVolume(), sceSSyn_SetOutPortVolume(),

  sceSSyn_SetOutputAssign(), sceSSyn_SetOutputMode(), sceSSyn_SetPortMaxPoly(),

  sceSSyn_SetPortVolume(), sceSSyn_SetTvaEnvMode()

Chapter 4: CSL Line-out for EE

- In sceLout_ATick() and sceLout_Init(), the "Calling conditions" descriptions have been corrected.
  (There is no change in function implementation).

Chapter 6: Standard Kit Library/Sound System

- In sceSkInit() and sceSkSsInit(), errors in the argument types have been corrected.

Chapter 7: SPU2 Local Memory Management Library

- In the following functions, the "Calling Conditions" descriptions have been corrected. (There is no change in function implementation.)

  sceSpu2MemAllocate()

  sceSpu2MemFree()

  sceSpu2MemInit()

  sceSpu2MemQuit()

Related Documentation

Library specifications for the IOP can be found in the *PlayStation®2 IOP Library Reference* manuals and the *PlayStation®2 IOP Library Overview* manuals.

Note: the Developer Support Web site posts current developments regarding the Libraries and also provides notice of future documentation releases and upgrades.

Typographic Conventions

Certain Typographic Conventions are used throughout this manual to clarify the meaning of the text:

Convention	Meaning
`courier`	Indicates literal program code.
italic	Indicates names of arguments and structure members (in structure/function definitions only).
medium bold	Indicates data types and structure/function names (in structure/function definitions only).
blue	Indicates a hyperlink.

Developer Support

Sony Computer Entertainment America (SCEA)

SCEA developer support is available to licensees in North America only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
In North America:	*In North America:*
Attn: Developer Tools Coordinator Sony Computer Entertainment America 919 East Hillsdale Blvd. Foster City, CA 94404, U.S.A. Tel: (650) 655-8000	E-mail: PS2_Support@playstation.sony.com Web: http://www.devnet.scea.com/ Developer Support Hotline: (650) 655-5566 (Call Monday through Friday, 8 a.m. to 5 p.m., PST/PDT)

Sony Computer Entertainment Europe (SCEE)

SCEE developer support is available to licensees in Europe only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
In Europe:	*In Europe:*
Attn: Production Coordinator Sony Computer Entertainment Europe 30 Golden Square London W1F 9LD, U.K. Tel: +44 (0) 20 7859-5000	E-mail: ps2_support@scee.net Web: https://www.ps2-pro.com/ Developer Support Hotline: +44 (0) 20 7859-5777 (Call Monday through Friday, 9 a.m. to 6 p.m., GMT)

Chapter 1: CSL MIDI Stream Generation
Table of Contents

Structures **1-3**
sceMSInHsMsg 1-3

Functions **1-4**
sceMSIn_Init 1-4
sceMSIn_MakeHsExpression 1-5
sceMSIn_MakeHsMsg1 1-6
sceMSIn_MakeHsMsg2 1-7
sceMSIn_MakeHsNoteOff 1-8
sceMSIn_MakeHsNoteOn 1-9
sceMSIn_MakeHsPanpot 1-10
sceMSIn_MakeHsPitchBend 1-11
sceMSIn_MakeHsPreExpression 1-12
sceMSIn_MakeHsPrePanpot 1-13
sceMSIn_MakeHsPrePitchBend 1-14
sceMSIn_MakeMsg/sceMSIn_MakeMsg3 1-15
sceMSIn_MakeMsg2 1-16
sceMSIn_NoteOff 1-17
sceMSIn_NoteOn 1-18
sceMSIn_NoteOnEx 1-19
sceMSIn_ProgramChange 1-20
sceMSIn_PutExcMsg 1-21
sceMSIn_PutHsMsg 1-22
sceMSIn_PutMsg 1-23

Structures

sceMSInHsMsg

Extended MIDI message

Library	*Introduced*	*Documentation last modified*
emsin	1.3	January 4, 2002

Structure

typedef struct {

unsigned char *d[7];*	Extended MIDI message

} sceMSInHsMsg;

Description

This structure is used for extended MIDI messages.

Functions

sceMSIn_Init

Initialization.

Library	*Introduced*	*Documentation last modified*
emsin	1.3	January 4, 2002

Syntax

int sceMSIn_Init(	
sceCslCtx **module_context***)**	Module Context address

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Only checks the validity of the module context.

Return value

When processing is successful: 0

sceMSIn_MakeHsExpression

Create extended Expression (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	March 26, 2001

Syntax

void sceMSIn_MakeHsExpression(

sceMSInHsMsg **hs_message,*	Extended MIDI message address
unsigned char *ch,*	Channel
unsigned char *key,*	Key number
unsigned char *id,*	ID number
unsigned char *expression***)**	Expression Data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

If *hs_message* does not conflict:

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This is a macro for creating an Expression Message of an extended Voice Control Message.

Return value

None

sceMSIn_MakeHsMsg1

Create extended Pre Voice Control Message (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	March 26, 2001

Syntax

void sceMSIn_MakeHsMsg1(	
sceMSInHsMsg **hs_message,*	Extended MIDI message address
unsigned char *op_code,*	Instruction code
unsigned char *ch,*	Channel
unsigned char *1st_data,*	Instruction-dependent data
unsigned char *2nd_data***)**	Instruction-dependent data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

If *hs_message* does not conflict:

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This is a macro for creating an extended Pre Voice Control Message.

Return value

None

sceMSIn_MakeHsMsg2

Create extended Voice Control Message (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	March 26, 2001

Syntax

void sceMSIn_MakeHsMsg2(

sceMSInHsMsg **hs_message,*	Extended MIDI message address
unsigned char *op_code,*	Instruction code
unsigned char *ch,*	Channel
unsigned char *key,*	Key number
unsigned char *id,*	ID number
unsigned char *1st_data,*	Instruction-dependent data
unsigned char *2nd_data***)**	Instruction dependent data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

If *hs_message* does not conflict:

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This is a macro for creating an extended Voice Control Message.

Return value

None

sceMSIn_MakeHsNoteOff

Create extended Note Off (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	March 26, 2001

Syntax

void sceMSIn_MakeHsNoteOff(

sceMSInHsMsg **hs_message,*	Extended MIDI message address
unsigned char *ch,*	Channel
unsigned char *key,*	Key number
unsigned char *id***)**	ID number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

If *hs_message* does not conflict:

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This is a macro for creating a Note Off Message of an extended Voice Control Message.

Return value

None

sceMSIn_MakeHsNoteOn

Create extended Note On (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	March 26, 2001

Syntax

void sceMSIn_MakeHsNoteOn(

sceMSInHsMsg **hs_message,*	Extended MIDI message address
unsigned char *ch,*	Channel
unsigned char *key,*	Key number
unsigned char *id,*	ID number
unsigned char *velocity***)**	velocity Data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

If *hs_message* does not conflict:

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This is a macro for creating a Note On Message of an extended Voice Control Message.

Return value

None

sceMSIn_MakeHsPanpot

Create extended Panpot (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	March 26, 2001

Syntax

void sceMSIn_MakeHsPanpot(	
sceMSInHsMsg **hs_message,*	Extended MIDI message address
unsigned char *ch,*	Channel
unsigned char *key,*	Key number
unsigned char *id,*	ID number
unsigned char *panpot***)**	Panpot Data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

If *hs_message* does not conflict:

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This is a macro for creating a Panpot Message of an extended Voice Control Message.

Return value

None

sceMSIn_MakeHsPitchBend

Create extended Pitch Bend (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	March 26, 2001

Syntax

void sceMSIn_MakeHsPitchBend(

sceMSInHsMsg **hs_message,*	Extended MIDI message address
unsigned char *ch,*	Channel
unsigned char *key,*	Key number
unsigned char *id,*	ID number
unsigned char *lsb_data,*	Pitch Bend LSB Data
unsigned char *msb_data***)**	Pitch Bend MSB Data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

If *hs_message* does not conflict:

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This is a macro for creating a Pitch Bend Message of an extended Voice Control Message.

Return value

None

sceMSIn_MakeHsPreExpression

Create extended Pre Expression (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	March 26, 2001

Syntax

void sceMSIn_MakeHsPreExpression(	
sceMSInHsMsg **hs_message,*	Extended MIDI message address
unsigned char *ch,*	Channel
unsigned char *expression***)**	Expression Data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

If *hs_message* does not conflict:

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This is a macro for creating an Expression Message of an extended Pre Voice Control Message.

Return value

None

sceMSIn_MakeHsPrePanpot

Create extended Pre Panpot (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	March 26, 2001

Syntax

void sceMSIn_MakeHsPrePanpot(

sceMSInHsMsg **hs_message,*	Extended MIDI message address
unsigned char *ch,*	Channel
unsigned char *panpot***)**	Panpot Data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

If *hs_message* does not conflict:

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This is a macro for creating a Panpot Message of an extended Pre Voice Control Message.

Return value

None

sceMSIn_MakeHsPrePitchBend

Create extended Pre Pitch Bend (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	March 26, 2001

Syntax

void sceMSIn_MakeHsPrePitchBend(	
sceMSInHsMsg **hs_message,*	Extended MIDI message address
unsigned char *ch,*	Channel
unsigned char *lsb_data,*	Pitch Bend LSB Data
unsigned char *msb_data***)**	Pitch Bend MSB Data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

If *hs_message* does not conflict:

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This is a macro for creating a Pitch Bend Message of an extended Pre Voice Control Message.

Return value

None

sceMSIn_MakeMsg/sceMSIn_MakeMsg3

Pack MIDI message into unsigned int (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	March 26, 2001

Syntax

unsigned int sceMSIn_MakeMsg(	
unsigned int *status,*	MIDI status
unsigned int *1st_data_byte,*	MIDI 1st data byte
unsigned int *2nd_data_byte***);**	MIDI 2nd data byte
unsigned int sceMSIn_MakeMsg3(	
unsigned int *status,*	MIDI status
unsigned int *1st_data_byte,*	MIDI 1st data byte
unsigned int *2nd_data_byte***);**	MIDI 2nd data byte

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Packs a MIDI message into an unsigned int.

The return value is used as an argument of sceMSIn_MakeMsg.

Return value

Packed MIDI message

sceMSIn_MakeMsg2

Pack MIDI message into unsigned int (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	March 26, 2001

Syntax

unsigned int sceMSIn_MakeMsg2(

unsigned int *status,*	MIDI status
unsigned int *1st_data_byte***)**	MIDI 1st data byte

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Packs a MIDI message into an unsigned int.

The return value is used as an argument of sceMSIn_MakeMsg.

The result is the same as when sceMSin_MakeMsg is used for a MIDI message with no *2nd_data_byte* or when the *2nd_data_byte* == 0 in sceMSIn_MakeMsg3.

Return value

Packed MIDI message

sceMSIn_NoteOff

Write a note-off message to the output stream (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	January 4, 2002

Syntax

int sceMSIn_NoteOff(

sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	output port number
unsigned int *midi_ch,*	MIDI channel
unsigned int *key_number***)**	Note number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes a note-off message to the specified output port buffer.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceMSIn function calls or sceHSyn_ATick() calls.

Return value

When processing is successful: 0

sceMSIn_NoteOn

Write a note-on message to the output stream (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	Januar y 4, 2002

Syntax

int sceMSIn_NoteOn(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Output port number
unsigned int *midi_ch,*	MIDI channel
unsigned int *key_number,*	Note number
unsigned int *velocity***)**	Velocity (strength of key strike)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes a note-on message to the specified output port buffer.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceMSIn function calls or sceHSyn_ATick() calls.

Return value

When processing is successful: 0

sceMSIn_NoteOnEx

Write a note-on message to the output stream (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	January 4, 2002

Syntax

int sceMSIn_NoteOnEx(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Output port number
unsigned int *midi_ch,*	MIDI channel
unsigned int *key_number,*	Note number
unsigned int *velocity,*	Velocity (strength of key strike)
unsigned int *prg_number***)**	Program number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes a program-change and a note-on message to the specified output port buffer.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceMSIn function calls or sceHSyn_ATick() calls.

Return value

When processing is successful: 0

sceMSIn_ProgramChange

Write a program-change message to the output stream (MACRO)

Library	*Introduced*	*Documentation last modified*
emsin	1.3	January 4, 2002

Syntax

int sceMSIn_ProgramChange(

sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Output port number
unsigned int *midi_ch,*	MIDI channel
unsigned int *prg_number***)**	Program number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes a program-change message to the specified output port buffer.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceMSIn function calls or sceHSyn_ATick() calls.

Return value

When processing is successful: 0

sceMSIn_PutExcMsg

Write exclusive message to output stream

Library	*Introduced*	*Documentation last modified*
emsin	1.3	January 4, 2002

Syntax

int sceMSIn_PutExcMsg(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Output port number
unsigned char **exc_data_addr,*	Exclusive data address
unsigned int *exc_data_length*)	Exclusive data size

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes an exclusive message to the specified output port buffer. Exclusive data must begin with 0xF0 and end with 0xF7.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceMSIn function calls or sceHSyn_ATick() calls.

Return value

When processing is successful: 0

sceMSIn_PutHsMsg

Write extended MIDI message to output stream

Library	*Introduced*	*Documentation last modified*
emsin	1.3	January 4, 2002

Syntax

int sceMSIn_PutHsMsg(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Output port number
sceMSInHsMsg **hs_message*)	Extended MIDI message address

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes an extended MIDI message to the specified output port buffer.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceMSIn function calls or sceHSyn_ATick() calls.

Return value

When processing is successful: 0

sceMSIn_PutMsg

Write MIDI message to output stream

Library	*Introduced*	*Documentation last modified*
emsin	1.3	January 4, 2002

Syntax

int sceMSIn_PutMsg(

sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Output port number
unsigned int *midi_message***)**	MIDI message: bits 0-7: status bits 8-15: 1st data byte bits 16-23: 2nd data byte

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes a MIDI message to the specified output port buffer.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceMSIn function calls or sceHSyn_ATick() calls.

Return value

When processing is successful: 0.

Chapter 2: CSL SE Stream Generation (for EE)
Table of Contents

Functions **2-3**

sceSEIn_ATick 2-3
sceSEIn_Init 2-4
sceSEIn_Load 2-5
sceSEIn_MakeAllNoteOff 2-6
sceSEIn_MakeAllNoteOffMask 2-7
sceSEIn_MakeAmpLFO 2-8
sceSEIn_MakeMsg / sceSEIn_MakeMsg4 2-9
sceSEIn_MakeNoteOn 2-10
sceSEIn_MakeNoteOnZero 2-11
sceSEIn_MakePitchLFO 2-12
sceSEIn_MakePitchOn 2-13
sceSEIn_MakePitchOnZero 2-14
sceSEIn_MakeTimePanpot 2-15
sceSEIn_MakeTimePitch 2-16
sceSEIn_MakeTimeVolume 2-17
sceSEIn_NoteOff 2-18
sceSEIn_NoteOn 2-19
sceSEIn_PitchOn 2-20
sceSEIn_PutMsg 2-21
sceSEIn_PutSEMsg 2-22

Functions

sceSEIn_ATick

Process interrupt

Library	*Introduced*	*Documentation last modified*
modsein	2.1	March 26, 2001

Syntax

int sceSEIn_ATick(

sceCslCtx **module_context)*	Address of Module Context

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function performs processing for each tick.

This is only a formal definition. No real processing is performed.

Return value

If processing was successful	0

sceSEIn_Init

Initialize

Library	*Introduced*	*Documentation last modified*
modsein	2.1	March 26, 2001

Syntax

int sceSEIn_Init (	
sceCslCtx **module_context)*	Address of Module Context

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function checks for a proper module context.

Return value

If processing was successful	0

sceSEIn_Load

Load data

Library	*Introduced*	*Documentation last modified*
modsein	2.1	March 26, 2001

Syntax

int sceSEIn_Load (	
sceCslCtx **module_context)*	Address of Module Context

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This is only a formal definition. No real processing is performed.

Return value

If processing was successful	0

sceSEIn_MakeAllNoteOff

Write All Note Off message to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.4	October 11, 2001

Syntax

int sceSEIn_MakeAllNoteOff (	
sceCslCtx **module_context*,	Address of Module Context
unsigned int *port_number*,	Output port number
unsigned int *id*)	SE message ID

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

This function writes an All Note Off message to the specified output port buffer.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with sceHSyn_Atick() or other sceSEIn functions.

Return value

When processing is successful, zero is returned.

sceSEIn_MakeAllNoteOffMask

Write All Note Off Mask message to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.4	October 11, 2001

Syntax

int sceSEIn_MakeAllNoteOffMask (	
sceCslCtx ******module_context***,**	Address of Module Context
unsigned int *port_number***,**	Output port number
unsigned int *id***,**	SE message ID
unsigned int *base_id***,**	Target base ID
unsigned int *mask***)**	Mask

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

This function writes an All Note Off Mask message to the specified output port buffer.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with sceHSyn_Atick() or other sceSEIn functions.

Return value

When processing is successful, zero is returned.

sceSEIn_MakeAmpLFO

Write amp LFO message to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.2	December 3, 2001

Syntax

int sceSEIn_MakeAmpLFO (	
sceCslCtx **module_context,*	Address of the Module Context
unsigned int *port_number,*	Output port number
unsigned int *id,*	SE message ID
unsigned int *bank_number,*	Bank number or sound effect timbre set number
unsigned int *prog_number,*	Program number or sound effect timbre number
unsigned int *note_number,*	Note number
unsigned int *depth_cycle,*	Amplitude or period
unsigned int *command***)**	Command function sceSEMsg_VCTRL_AMPLFO_DEPTH_P Sets the positive amplitude of the amp LFO sceSEMsg_VCTRL_AMPLFO_DEPTH_M Sets the negative amplitude of the amp LFO sceSEMsg_VCTRL_AMPLFO_CYCLE Sets the period of the amp LFO

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes an amp LFO message to the specified output port buffer.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with sceHSyn_Atick() or other sceSEIn functions.

Return value

0 if successful

sceSEIn_MakeMsg / sceSEIn_MakeMsg4

Pack SE message into an unsigned int (MACRO)

Library	*Introduced*	*Documentation last modified*
modsein	2.1	March 26, 2001

Syntax

unsigned int sceSEIn_MakeMsg (

unsigned int *status,*	SE status
unsigned int *1st_data_byte,*	SE 1st data byte
unsigned int *2nd_data_byte,*	SE 2nd data byte
unsigned int *3rd_data_byte)*	SE 3rd data byte

unsigned int sceSEIn_MakeMsg4 (

unsigned int *status,*	SE status
unsigned int *1st_data_byte,*	SE 1st data byte
unsigned int *2nd_data_byte,*	SE 2nd data byte
unsigned int *3rd_data_byte)*	SE 3rd data byte

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function packs an SE message into an unsigned int.

The return value is used as an argument for sceSEIn_PutMsg.

Return value

Packed SE message

sceSEIn_MakeNoteOn

Write Note On/Off message to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.2	December 3, 2001

Syntax

int sceSEIn_MakeNoteOn (

sceCslCtx **module_context,*	Address of the Module Context
unsigned int *port_number,*	Output port number
unsigned int *id,*	SE message ID
unsigned int *bank_number,*	Bank number or sound effect timbre set number
unsigned int *prog_number,*	Program number
unsigned int *note_number,*	Note number
unsigned int *velocity,*	Velocity (keypress intensity)
int *panpot***)**	Panpot

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes a Note On/Off message to the specified output port buffer.

A velocity of 0 is handled as a Note Off.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with sceHSyn_Atick() or other sceSEIn functions.

Return value

0 if successful

sceSEIn_MakeNoteOnZero

Write Note On message to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.4.2	December 3, 2001

Syntax

int sceSEIn_MakeNoteOnZero (

sceCslCtx **module_context*,	Address of Module Context
unsigned int *port_number*,	Output port number
unsigned int *id*,	SE message ID
unsigned int *bank_number*,	Bank number or sound effect timbre set number
unsigned int *prog_number*,	Program number
unsigned int *note_number*,	Note number
unsigned int *velocity*,	Velocity (key strike intensity)
int *panpot*)	Panpot

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

This function writes a Note On message to the specified output port buffer. When velocity is 0, the voice is still assigned and sound is generated but with volume 0. To mute this sound, use sceSEIn_MakeNoteOn().

This function can be called in a multithreaded environment from an interrupt-enabled state, if it does not conflict with other sceSEIn functions and sceHSyn_ATick().

Return value

When processing is successful	0

sceSEIn_MakePitchLFO

Write pitch LFO message to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.2	December 3, 2001

Syntax

int sceSEIn_MakePitchLFO (	
sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Output port number
unsigned int *id,*	SE message ID
unsigned int *bank_number,*	Bank number or sound effect timbre set number
unsigned int *prog_number,*	Program number or sound effect timbre number
unsigned int *note_number,*	Note number
unsigned int *depth_cycle,*	Amplitude or period
unsigned int *command***)**	Command function sceSEMsg_VCTRL_PITCHLFO_DEPTH_P Sets the positive amplitude of the pitch LFO sceSEMsg_VCTRL_PITCHLFO_DEPTH_M Sets the negative amplitude of the pitch LFO sceSEMsg_VCTRL_PITCHLFO_CYCLE Sets the period of the pitch LFO

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes a pitch LFO message to the specified output port buffer.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with sceHSyn_Atick() or other sceSEIn functions.

Return value

0 if successful

sceSEIn_MakePitchOn

Write Note On/Off message (specified pitch) to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.2	December 3, 2001

Syntax

int sceSEIn_MakePitchOn (	
sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Output port number
unsigned int *id,*	SE message ID
unsigned int *bank_number,*	Bank number or sound effect timbre set number
unsigned int *prog_number,*	Program number or sound effect timbre number
unsigned int *note_number,*	Note number
unsigned int *velocity,*	Velocity (keypress intensity)
int *panpot,*	Panpot
unsigned int *pitch***)**	Generated pitch

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes Note On/Off message (specified pitch) to the specified output port buffer.

A velocity of 0 is handled as a Note Off.

Pitch is a value specified by SD_VP_PITCH (0 ~ 0x3fff) in the low-level sound library.

In the current implementation, if sound generation is performed using this function, then the specification of PitchLFO in the bank binary data will be made ineffective.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with sceHSyn_Atick() or other sceSEIn functions.

Notes

In the current implementation, Time-Pitch cannot be performed for a voice if sound was generated by either a Note On/Off message (with pitch specification) or a Note on message (with pitch specification). To generate sound which will perform Time-Pitch processing, a Note On/Off message or Note on message must be used.

Return value

0 if successful

sceSEIn_MakePitchOnZero

Write Note On message (with pitch specification) to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.4.2	December 3, 2001

Syntax

int sceSEIn_MakePitchOnZero (

sceCslCtx **module_context*,	Address of Module Context
unsigned int *port_number*,	Output port number
unsigned int *id*,	SE message ID
unsigned int *bank_number*,	Bank number or sound effect timbre set number
unsigned int *prog_number*,	Program number or sound effect timbre number
unsigned int *note_number*,	Note number
unsigned int *velocity*,	Velocity (key strike intensity)
int *panpot*,	Panpot
unsigned int *pitch***)**	Generated pitch

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

This function writes a Note On message (with pitch specification) to the specified output port buffer. When velocity is 0, the voice is still assigned and sound is generated, but with volume 0. To mute this sound, use sceSEIn_MakePitchOn().

pitch is the value that is specified by SD_VP_PITCH in the low-level sound library (range is 0-0x3fff).

In the current implementation, when sound is generated by this function, the PitchLFO specification in the bank binary data is ignored.

This function can be called in a multithreaded environment from an interrupt-enabled state, if it does not conflict with other sceSEIn functions and sceHSyn_ATick().

Notes

In the current implementation, Time-Pitch cannot be performed for a voice if sound was generated by either a Note On/Off message (with pitch specification) or a Note On message (with pitch specification). To generate sound that will perform Time-Pitch processing, a Note On/Off message or Note On message must be used.

Return value

When processing is successful	0

sceSEIn_MakeTimePanpot

Write time pan pot message to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.2	December 3, 2001

Syntax

int sceSEIn_MakeTimePanpot (	
sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Output port number
unsigned int *id,*	SE message ID
unsigned int *bank_number,*	Bank number or sound effect timbre set number
unsigned int *prog_number,*	Program number or sound effect timbre number
unsigned int *note_number,*	Note number
unsigned int *delta_time,*	Elapsed time (units : milliseconds)
int *target_panpot,*	Target panpot
unsigned int *command***)**	Command function sceSEMsg_VCTRL_TIME_PANPOT_CW Moves the panpot in the clockwise direction sceSEMsg_VCTRL_TIME_PANPOT_CCW Moves the panpot in the counter-clockwise direction

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes a time pan pot message to the specified output port buffer.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with sceHSyn_Atick() or other sceSEIn functions.

Return value

0 if successful

sceSEIn_MakeTimePitch

Write time pitch message to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.2	October 11, 2001

Syntax

int sceSEIn_MakeTimePitch (	
sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Output port number
unsigned int *id,*	SE message ID
unsigned int *bank_number,*	Bank number or sound effect timbre set number
unsigned int *prog_number,*	Program number or sound effect timbre number
unsigned int *note_number,*	Note number
unsigned int *delta_time,*	Elapsed time (units: milliseconds)
unsigned int *target_pitch,*	Target pitch (units: cents)
unsigned int *command***)**	Command function sceSEMsg_VCTRL_TIME_PITCH_P Raises the pitch sceSEMsg_VCTRL_TIME_PITCH_M Lowers the pitch

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes a time pitch message to the specified output port buffer.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with sceHSyn_Atick() or other sceSEIn functions.

Return value

0 if successful.

sceSEIn_MakeTimeVolume

Write time volume message to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.2	December 3, 2001

Syntax

int sceSEIn_MakeTimeVolume (

sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Output port number
unsigned int *id,*	SE message ID
unsigned int *bank_number,*	Bank number or sound effect timbre set number
unsigned int *prog_number,*	Program number or sound effect timbre number
unsigned int *note_number,*	Note number
unsigned int *delta_time,*	Elapsed time (units: milliseconds)
unsigned int *target_volume***)**	Target volume

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes a time volume message to the specified output port buffer.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with sceHSyn_Atick() or other sceSEIn functions.

Return value

0 if successful.

sceSEIn_NoteOff

Writes a message which performs Note Off to output port buffer (MACRO)

Library	*Introduced*	*Documentation last modified*
modsein	2.1	December 3, 2001

Syntax

int sceSEIn_NoteOff (	
sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Output port number
unsigned int *id,*	SE message ID
unsigned int *bank_number,*	Bank number or sound effect timbre set number
unsigned int *prog_number,*	Program number or sound effect timbre number
unsigned int *note_number***)**	Note number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

This function writes a message which performs Note Off (i.e. uses a Note On/Off message, velocity=0) to the specified output port buffer.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with sceHSyn_Atick() or other sceSEIn functions.

Return value

If processing was successful	0

sceSEIn_NoteOn

Writes a message which performs Note On/Off to output port buffer (MACRO)

Library	*Introduced*	*Documentation last modified*
modsein	2.1	December 3, 2001

Syntax

int sceSEIn_NoteOn (

sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Output port number
unsigned int *id,*	SE message ID
unsigned int *bank_number,*	Bank number or sound effect timbre set number
unsigned int *prog_number,*	Program number or sound effect timbre number
unsigned int *note_number,*	Note number
unsigned int *velocity,*	Velocity (key strike intensity)
int *panpot***)**	Panpot

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

This function writes a message which performs Note On/Off (i.e. uses a Note On/Off message) to the specified output port buffer. When velocity is 0, the message is handled as a Note Off message.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with sceHSyn_Atick() or other sceSEIn functions.

Return value

If processing was successful	0

sceSEIn_PitchOn

Write Note On/Off message (pitch specification) to output port buffer (MACRO)

Library	*Introduced*	*Documentation last modified*
modsein	2.1	December 3, 2001

Syntax

int sceSEIn_PitchOn (

sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Output port number
unsigned int *id,*	SE message ID
unsigned int *bank_number,*	Bank number or sound effect timbre set number
unsigned int *prog_number,*	Program number or sound effect timbre number
unsigned int *note_number,*	Note number
unsigned int *velocity,*	Velocity (key strike intensity)
int *panpot,*	Panpot
unsigned int *pitch)*	Generated pitch

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

This function writes a message which performs Note On/Off (pitch specification) (i.e. uses a Note On/Off message) to the specified output port buffer.

The pitch is the value (0 to 0x3fff) that is specified by SD_VP_PITCH in the low level sound library.

In the current implementation, when sound is generated by this function, the PitchLFO specification in the bank binary data will become invalid.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with sceHSyn_Atick() or other sceSEIn functions.

Return value

If processing was successful	0

sceSEIn_PutMsg

Write SE Message to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.1	October 11, 2001

Syntax

int sceSEIn_PutMsg (

sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Output port number
unsigned int *id,*	SE message ID
unsigned int *se_msg1,*	SE message
	bit 0-7: SE status
	bit 8-15: 1st data byte
	bit 16-23: 2nd data byte
	bit 24-31: 3rd data byte
unsigned int *se_msg2)*	SE message
	bit 0-7: 4th data byte
	bit 8-15: 5th data byte
	bit 16-23: 6th data byte
	bit 24-31: 7th data byte

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

This function writes an SE message to the specified output port buffer.

This function only supports SE messages for which the SE status is 0xa?.

For writing an arbitrary SE message, use sceSEIn_PutSEMsg().

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with sceHSyn_Atick() or other sceSEIn functions.

Return value

If processing was successful	0

sceSEIn_PutSEMsg

Write arbitrary SE message to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.2	October 11, 2001

Syntax

int sceSEIn_PutSEMsg (

sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Output port number
unsigned int *id,*	SE message ID
unsigned char **msg,*	Address of the buffer that contains the SE message
unsigned int *msg_length*)	Length of the SE message within the buffer specified by msg. (units: bytes)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in interrupt-disabled state)

Description

Writes an SE message to the specified output port buffer.

The contents of the msg are the SE status and SE data of the SE message.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with sceHSyn_Atick() or other sceSEIn functions.

Return value

0 if successful

Chapter 3: CSL Software Synthesizer
Table of Contents

Structures **3-3**
sceSSynChOutAttrib 3-3
sceSSynConf 3-4
sceSSynEnv 3-5
sceSSynNrpnMsg 3-6
sceSSynRpnMsg 3-7

Functions **3-8**
sceSSyn_ATick 3-8
sceSSyn_BreakAtick 3-9
sceSSyn_ClearBreakAtick 3-10
sceSSyn_Init 3-11
sceSSyn_Load 3-12
sceSSyn_PrepareParameter 3-13
sceSSyn_RegisterRpc 3-14
sceSSyn_SendExcMsg 3-15
sceSSyn_SendNrpnMsg 3-16
sceSSyn_SendRpnMsg 3-17
sceSSyn_SendShortMsg 3-18
SceSSyn_SetChPriority 3-19
sceSSyn_SetMasterVolume 3-20
sceSSyn_SetOutPortVolume 3-21
sceSSyn_SetOutputAssign 3-22
sceSSyn_SetOutputMode 3-23
sceSSyn_SetPortMaxPoly 3-24
sceSSyn_SetPortVolume 3-25
sceSSyn_SetTvaEnvMode 3-26

Callback Functions **3-27**
exc_callback 3-27
msg_callback 3-28

Structures

sceSSynChOutAttrib

Specify output mode for each input channel

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	July 24, 2000

Structure

typedef struct {	
unsigned char *ch;*	Channel for which mode is to be set
unsigned char *ch_output;*	Output port within channel
unsigned char *mode;*	Output mode sceSSynMuteOut: No output sceSSynMonoOut: Mono output sceSSynLOut: Left channel output sceSSynROut: Right channel output
unsigned char *output_line;*	Number of PCM stream to be output
unsigned int *att;*	Attenuator SSYN_VOLUME_0DB: 0db
} sceSSynChOutAttrib;	

Description

This structure specifies output modes for input channels. It is specified as an argument of sceSSyn_SetOutputAssign().

sceSSynConf

Initial setting information

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	July 24, 2000

Structure

#define sceSSynVoiceBufferSize: 576

#define sceSSynInputBufferSize: (352*16)

typedef struct {

unsigned int *unit_samples;*	Number of data to be output in a single operation (units: samples)
unsigned int *sampling_frequency;*	Output PCM sampling rate
unsigned int *n_voices;*	Maximum number of voices for the entire system
void **voice_buffer;*	Voice status management buffer
unsigned int *voice_buffer_size;*	Voice status management buffer size sceSSynVoiceBufferSize * n_voices or higher is required
unsigned int *n_input_port;*	Number of input ports Same as the number of Buffer Group 0 buffer contexts
void * *input_port_buffer;*	Input management buffer
unsigned int *input_port_buffer_size;*	Input management buffer size sceSSynInputBufferSize * n_input_port or higher is required

} sceSSynConf;

Description

This structure specifies the initial setting information. It is allocated to the config member of the CSL context structure sceCslCtx.

sceSSynEnv

Input environment

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	July 24, 2000

Structure

typedef struct {	
unsigned int *input_buff_len;*	Size of buffer for communicating with the IOP. This must be an integer multiple of 16.
void **input_buff;*	Buffer for communicating with the IOP. Since non-cached access is performed, care should be taken with alignment.
void **tone_param;*	Pointer to tone data (unused)
unsigned int *(*msg_callback)* **(unsigned int, unsigned int);**	MIDI message filter callback address
unsigned int *msg_callback_private_data;*	User data that is passed as a MIDI message filter callback argument.
unsigned int *(*exc_callback)***(unsigned int, unsigned char*, unsigned int, unsigned char*, unsigned int);**	MIDI exclusive message filter callback address
unsigned int *exc_callback_private_data;*	User data that is passed as a MIDI exclusive message filter callback argument.
unsigned int *system* **[(sceSSynEnvSize+sizeof(int)-1)/sizeof(int)];**	Internal variable argument. To not use IOP MIDI Stream data, have input_buff_len=0; input_buff=NULL; set by a port, and connection requests for that port from the IOP will be refused.
} sceSSynEnv;	

Description

This structure is used in the input buffer which controls the software synthesizer input, playback state, etc.

Notes

To not use IOP MIDI Stream data, have input_buff_len=0; input_buff=NULL; set by a port, and connection requests for that port from the IOP will be refused.

sceSSynNrpnMsg

NRPN transmission structure

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	July 24, 2000

Structure

typedef struct {	
unsigned char *ch;*	MIDI channel
unsigned short *num;*	NRPN number
unsigned short *data;*	NRPN data
} sceSSynNrpnMsg;	

Description

NRPN transmission structure.

sceSSynRpnMsg

RPN transmission structure

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	July 24, 2000

Structure

typedef struct {	
unsigned char *ch;*	MIDI channel
unsigned short *num;*	RPN number
unsigned short *data;*	RPN data
} sceSSynRpnMsg;	

Description

RPN transmission structure.

Functions

sceSSyn_ATick

Process interrupts

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	January 4, 2002

Syntax

int sceSSyn_ATick(

sceCslCtx **module_context***)**	Module Context address

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Reads data from the IOP and generates PCM Stream Data.

Return value

When processing is successful: 0

sceSSyn_BreakAtick

Interrupt sceSSyn_ATick() processing

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	January 4, 2002

Syntax

int sceSSyn_BreakAtick(	
sceCslCtx **module_context)*	Module Context address

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

Interrupts the current sceSSyn_ATick() process that is executing. Can be called from an interrupt.

Processing is interrupted once an individual process (4 voices) ends.

Also, after the process is interrupted, the output buffer is cleared and any voice generation in progress is terminated.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

Normal termination: 0

sceSSyn_ClearBreakAtick

Cancel interruption of sceSSyn_ATick() processing

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	January 4, 2002

Syntax

int sceSSyn_ClearBreakAtick(

sceCslCtx **module_context)*	Module Context address

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

Can be called from an interrupt.

After sceSSyn_BreakAtick() has been called, sceSSyn_ATick() processing will be interrupted until sceSSyn_ClearBreakAtick() is called.

sceSSyn_ClearBreakAtick() is typically called at the start of time measurement. When the permitted time is exceeded due to an interrupt or other cause, processing can be interrupted by calling sceSSyn_BreakAtick().

sceSSyn_BreakAtick() should really be used only in an emergency. Satisfactory results that control the load can normally be obtained for the output voices by limiting the maximum number of generated voices for the entire device or for each port.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

Normal termination: 0

sceSSyn_Init

Initialize

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	January 4, 2002

Syntax

int sceSSyn_Init(

sceCslCtx **module_context,*	Module Context address
unsigned int *interval)*	Period in which Atick() will be called (in microseconds)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

Initializes internal variables.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

When processing is successful: 0

sceSSyn_Load

Register phoneme and parameter data

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	January 4, 2002

Syntax

int sceSSyn_Load(

sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
void **parameter*)	Phoneme and parameter data address

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

Registers phoneme and parameter data at an input port.

The sceSSyn_PrepareParameter() function must be used in advance to resolve the address for the data to be registered.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

When processing is successful: 0

sceSSyn_PrepareParameter

Resolve phoneme and parameter data address

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	January 4, 2002

Syntax

int sceSSyn_PrepareParameter(

void ******parameter,*	Phoneme and parameter data address
unsigned int *size***)**	Phoneme and parameter data size

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

Since the address encoding for the phoneme and parameter data that is saved on disk differs from that used within the Software Synthesizer, this sceSSyn_PrepareParameter() function must be used to convert the address format.

Also, the location in memory cannot be changed for data to which sceSSyn_PrepareParameter() has been applied.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

When processing is successful: 0

sceSSyn_RegisterRpc

Register RPC server to reserve channel for communication with IOP

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	March 26, 2001

Syntax

int sceSSyn_RegisterRpc(	
sceCslCtx **module_context,*	Module Context address
int *priority***)**	Server thread priority

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Starts up the RPC server to reserve a channel for communication with the IOP.

Not necessary if IOP MIDI Stream is not being used.

Return value

When processing is successful: 0

sceSSyn_SendExcMsg

Input MIDI exclusive message

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	March 26, 2001

Syntax

int sceSSyn_SendExcMsg(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned char *exc_*data,*	Exclusive data address Must begin with 0xF0 and end with 0xF7.
unsigned int *length***)**	Exclusive data size

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Currently not implemented

Description

This is an API for inputting a MIDI exclusive message that bypasses the IOP.

Return value

When processing is successful: 0

sceSSyn_SendNrpnMsg

Input MIDI NRPN message

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	March 26, 2001

Syntax

int sceSSyn_SendNrpnMsg(	
sceCslCtx *module_context,	Module Context address
unsigned int port_*number,*	Input port number
sceSSynNrpnMsg **nrpn***)**	NRPN message address

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Currently not implemented

Description

This is an API for inputting a MIDI NRPN message that bypasses the IOP.

Return value

When processing is successful: 0

sceSSyn_SendRpnMsg

Input MIDI RPN message

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	January 4, 2002

Syntax

int sceSSyn_SendRpnMsg(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
sceSSynRpnMsg **rpn*)	RPN message address

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

This is an API for inputting a MIDI RPN message that bypasses the IOP.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

When processing is successful: 0

sceSSyn_SendShortMsg

Input MIDI message

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	January 4, 2002

Syntax

int sceSSyn_SendShortMsg(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned int *message***)**	MIDI message bit 0-7: status 8-15: 1st data 16-23: 2nd data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

This is an API for inputting a MIDI message that bypasses the IOP.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

When processing is successful: 0

SceSSyn_SetChPriority

Set CH priority

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	January 4, 2002

Syntax

int sceSSyn_SetChPriority(	
sceCslCtx **module_context,*	Module Context address
unsigned int port_*number,*	Input port number
unsigned int *ch,*	MIDI channel
unsigned char *priority***)**	Priority

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

The minimum priority is 0 and the maximum priority is 255.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

When processing is successful: 0

sceSSyn_SetMasterVolume

Set master volume

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	January 4, 2002

Syntax

int sceSSyn_Load(

sceCslCtx **module_context,*	Module Context address
unsigned int *volume***)**	Master volume value SSYN_VOLUME_0DB: 0db

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

Sets the master volume.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

When processing is successful: 0

sceSSyn_SetOutPortVolume

PCM Stream volume

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	January 4, 2002

Syntax

int sceSSyn_SetOutPortVolume(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	PCM Stream number
unsigned int *vol***)**	Volume SSYN_VOLUME_0DB: 0db

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

Sets the volume of the PCM Stream having the specified number.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

When processing is successful: 0

sceSSyn_SetOutputAssign

Assign channel output

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	January 4, 2002

Syntax

int sceSSyn_SetOutputAssign(	
sceCslCtx **module_context,*	Module Context address
unsigned int port_*number,*	Input port number
sceSSynChOutAttrib **attrib*)	Assigned information address

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

Assigns channel output (4 channels) to a PCM Stream.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

When processing is successful: 0

sceSSyn_SetOutputMode

Switch output mode (monaural/stereo)

Library	*Introduced*	*Documentation last modified*
libssyn	1.4	January 4, 2002

Syntax

int sceSSyn_SetOutputMode(	
int *output_mode)*	Output mode: sceSSynOutputMode_Mono: Disables the panpot sceSSynOutputMode_Stereo: Enables the panpot

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

Sets the output mode (enables/disables the panpot).

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

When processing is successful: 0

sceSSyn_SetPortMaxPoly

Limit number of voices for individual input port

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	January 4, 2002

Syntax

int sceSSyn_SetPortMaxPoly(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned char *max_polyphony)*	Maximum number of simultaneous voices

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

Sets the upper limit of the number of voices that are generated simultaneously for each input port.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

When processing is successful: 0

sceSSyn_SetPortVolume

Set input port volume

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	January 4, 2002

Syntax

int sceSSyn_SetPortVolume(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned int *vol***)**	Volume SSYN_VOLUME_0DB: 0db

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

Sets the master volume of the sound source having the specified input port.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

When processing is successful: 0

sceSSyn_SetTvaEnvMode

Set TVA envelope (release) operating mode

Library	*Introduced*	*Documentation last modified*
libssyn	2.0	January 4, 2002

Syntax

int sceSSyn_ SetTvaEnvMode(	
int *env_mode)*	TVA envelope (release) operating mode sceSSynTvaEnvMode_Fixed Always set TVA release time to the time set in the parameter sceSSynTvaEnvMode_ChangeByLevel Change time set in the parameter according to current level (if current value is small, time is short, and time is long for large values) (default value, to maintain compatibility)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

This function sets the TVA envelope (release) operating mode.

This function can be called in a multithreaded environment in an interrupt-enabled state if it does not conflict with other sceSSyn function calls.

Return value

When processing is successful: 0

Callback Functions

exc_callback

MIDI exclusive message filter callback

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	December 23, 1999

Syntax

unsigned int exc_callback(

unsigned int *exc_callback_private_data,*	sceSSynEnv exc_callback_private_data
unsigned char **data1,*	Exclusive data address
unsigned int *length1,*	data1 data size
unsigned char **data2,*	Exclusive data address
unsigned int *length2***)**	data2 data size

Description

Used for a MIDI exclusive message filter.

Since the ring buffer is directly referenced, two addresses are held as arguments.

When the data goes beyond the endpoint of the ring buffer and continues at the beginning of the buffer, valid values are entered for *data2* and *length2*.

If the data exists in a contiguous area, *data2* == NULL and *length2* == 0 are guaranteed.

The data starting position is the data byte following the exclusive status (0xF0).

This callback can be used only if EXEC_CALLBACK was defined when the library was built.

Return value

If 0 is returned, no message was transmitted.

msg_callback

MIDI message filter callback

Library	*Introduced*	*Documentation last modified*
libssyn	1.1	December 23, 1999

Syntax

unsigned int msg_callback(

unsigned int *msg_callback_private_data,*	sceSSynEnv msg_callback_private_data
unsigned int *message***)**	MIDI message
	bit 0-7: status
	8-15: 1st data
	16-23: 2nd data

Description

Used for a MIDI message filter.

This callback can be used only if EXEC_CALLBACK was defined when the library was built.

Return value

MIDI message that is transmitted to the Synthesizer.

If 0 is returned, no message was transmitted.

Chapter 4: CSL Line-out for EE
Table of Contents

Structures **4-3**
sceLoutConf 4-3

Functions **4-4**
sceLout_ATick 4-4
sceLout_Init 4-5

Structures

sceLoutConf

Environment

Library	*Introduced*	*Documentation last modified*
liblout	1.5	January 4, 2002

Structure

typedef struct {

unsigned char *attrib;*	Attribute: specify sceLoutDmaPreWait sceLoutDmaPreWait: Once DMA transfer is begun, do not wait for the transfer to finish. When starting the next DMA transfer, check to see if finished and, if not, wait for completion.
unsigned char *lineAssign[sceLoutMaxLine];*	Assign input buffer for SPU2 output. Set up the input buffer number corresponding to each channel, as below. If not used, use sceLoutNoOutPut lineAssign[0] CH 0 L lineAssign[1] CH 0 R lineAssign[2] CH 1 L lineAssign[3] CH 1 R
unsigned int *iopBufSize;*	Size of the buffer on the IOP that will be used for transferring data for the SPU2. The buffer size must be at least 4 X (size of the input PCM buffer) and must be an integral multiple of sceLoutInputUnit. 4X is required because 2X is needed to perform L/R output for each channel and another 2X is needed because it is a double buffer).
void * *iopBuf***[2];**	Buffer address on the IOP for transferring data to the SPU2. iopBuf[0]: for CH 0 iopBuf[1]: for CH 1 No output is sent to a channel for which a buffer is NULL.The buffer size must be iopBufSize.
unsigned int *nDmaBuf;*	Number of buffer dma arrays for DMA Operation. Since multiple DMA transfers are started during an sceLout_ATick(), specify a maximum number of requests that can be queued in a single sceSifSetDma(). The value must be at least 1. If sceLoutDmaPreWait is not set, 1 is adequate. If sceLoutDmaPreWait is set, a larger value will reduce the wait time. Refer to SIF DMA for information on maximum value.
sceSifDmaData *dma***[0];**	DMA Operation buffer. Must be an array with nDmaBuf elements.

} sceLoutConf;

Description

Initializes liblout.

Functions

sceLout_ATick

Interrupt handling

Library	*Introduced*	*Documentation last modified*
liblout	1.2	January 4, 2002

Syntax

int sceLout_ATick(

sceCslCtx **module_context*)	Address of Module Context

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

If space is available in an IOP buffer, PCM Stream Data is DMA transferred to the IOP buffer.

Return value

When processing is successful: 0

sceLout_Init

Initialize

Library	*Introduced*	*Documentation last modified*
liblout	1.2	January 4, 2002

Syntax

int sceLout_Init(

sceCslCtx **module_context,*	Address of Module Context
unsigned int *interval)*	Indicates how often ATick() will be called (in microseconds)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Initializes internal variables, starts librsd Auto DMA transfer.

Return value

When processing is successful: 0

Chapter 5: Low-Level Sound Library
Table of Contents

Functions **5-3**
sceSdCallBack 5-3
sceSdRemote 5-4
sceSdRemoteCallbackInit 5-6
sceSdRemoteInit 5-7
sceSdTransToIOP 5-8

Functions

sceSdCallBack

Set end callback for non-blocking execution

Library	*Introduced*	*Documentation last modified*
libsdr	1.1	March 26, 2001

Syntax

sceSifEndFunc sceSdCallBack(

sceSifEndFunc *end_func***)**	Address of end callback function

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt-disabled state)

Description

Sets the callback function when sceSdRemote() is executed in non-blocking mode. The callback function will be executed in interrupt mode. No callback function is executed when sceSdRemote() is executed in blocking mode or when NULL is specified for end_func. The initial value is NULL.

Non-blocking processing is performed to improve EE efficiency. However, applying an end callback to the EE causes a context switch, and this is also linked to a decrease in processing efficiency. The most efficient technique is to perform non-blocking processing and to confirm the end of IOP-side processing by using polling, not a callback. The SD_WRITE_EE and SD_RETURN_EE batch commands can be used for this (see libsd). Since these only perform a SIF DMA transfer from the IOP side, no interrupt will be generated on the EE side.

Notes

Since the end func callback function is executed as an interrupt handlers, special care is required when programming. Refer to the “Interrupt Handler Descriptions” section of \overview\eekernel for details.

Return value

Address of end callback function that had just been set.

sceSdRemote

Remotely execute libsd command

Library	*Introduced*	*Documentation last modified*
libsdr	1.1	March 26, 2001

Syntax

int sceSdRemote(

int *is_block,*	Indicates whether or not to block the EE until IOP-side processing ends. 1: Block 0: Do not block.
int *command,*	Command
int *arg,*	Arguments for the command. Variable length.
...)	

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

The libsd.irx API on the IOP is remotely executed from the EE according to the command. The command is specified using the libsd function name with the "sce" part of the name replaced by "r". For details, refer to iop/include/sdrcmd.h.

Since the arguments are of variable length, the number required for the particular command (function) should be specified.

Example: sceSdRemote(1, rSdSetParam, SD_CORE_0|SD_P_MVOLL , 0x3fff);

The return value of sceSdRemote will be the return value of the specified command. However, if non-blocking is specified, the return value will always be 0. Since the argument and return value type are always int, the values should be appropriately cast.

When non-blocking is specified, control returns from the function without waiting for IOP-side processing to end. Use sceSdCallBack() to set the end callback function to find out when IOP-side processing has completed. If the next command is sent before IOP processing ends, an invalid operation may occur.

If the specified command is either rSdProcBatch or rSdProcBatchEx, the argument addresses will be treated as IOP-side addresses. The transfer of command arrays or return value arrays to (or from) the IOP should be performed independently. The SD_RETURN_EE batch command can be used to transfer the return value array (see libsd).

Since transfers are performed internally by libsdr for rSdSetEffectAttr and rSdGetEffectAttr, EE addresses can be specified directly for parameter structure pointers. Refer to the "Memory alignment in transferred data" section of the libsdr Library Overview.

A command that specifies a callback, rSdSetTransCallback, rSdSetIRQCallback, rSdSetTransIntrHandler and rSdSetSpu2IntrHandler or a function on the EE, can be specified as a callback, and that function will be called as a thread. However, sceSdRemoteCallbackInit() should be called first. The pointer of the data passed to rSdSetTransIntrHandler and rSdSetSpu2IntrHandler is also the EE address.

If the command is rSdChangeThreadPriority, take two arguments and specify the priority value of the sdrdrv main thread and callback thread running on the IOP in order.

There are two thread priorities in sdrdrv.irx--the first is the priority used for the main thread, and the second is the priority used for the callback thread. The thread priority values are both 24 by default.

The priority value of the callback thread must be greater than or equal to the priority value of the main thread (the callback thread must have a lower priority than the main thread).

When changing the priority of an IOP thread, care must be taken with regard to priorities of other modules.

It is not recommended that the priority of IOP threads be changed casually.

Return value

Return value appropriate for the command.

However, when 0 is specified for *is_block*, the return value is always 0.

sceSdRemoteCallbackInit

Initialize the libsd callback environment

Library	*Introduced*	*Documentation last modified*
libsdr	1.1	March 26, 2001

Syntax

int sceSdRemoteCallbackInit(

int *priority***)**	Priority of EE thread that is started up for callback. This must be set higher than the main thread.

Description

When an interrupt callback is specified by an sceSdRemote command (rSdSetTransCallback or rSdSetIRQCallback), this function should be called only once in advance.

Since an IOP-side callback function will be received on the EE, one thread will be created internally. Therefore, the callback operates in thread mode, not interrupt mode.

If another callback is called while a callback is executing, the second callback enters a queue and waits until the first callback ends.

Since the thread ID is returned, post-processing such as DeleteThread should be performed when a thread is no longer necessary.

Since a callback that is specified by sceSdCallBack(), which is related to the non-blocking execution of sceSdRemote(), is different than the command callback mentioned here, no initialization is required for this API.

Return value

ID of the EE thread that is started up for callback.

If an error occurs, a negative number is returned.

sceSdRemoteInit

Initialize libsd remote environment

Library	*Introduced*	*Documentation last modified*
libsdr	1.1	March 26, 2001

Syntax

int sceSdRemoteInit(void)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Initializes the libsd remote environment.

Return value

Normal termination: 0. If an error occurred: -1.

sceSdTransToIOP

Transfers buffer on the EE to IOP memory

Library	*Introduced*	*Documentation last modified*
libsdr	1.5	July 2, 2001

Syntax

int sceSdTransToIOP(

void **buff,*	Location of transfer source in EE memory
u_int *sendAddr,*	Location of transfer destination in IOP memory
u_int *size,*	Size
u_int *isBlock)*	Determines whether or not to block the EE until IOP processing is complete. Blocks if 1 and does not block if 0.

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Transfers data in EE memory to IOP memory.

When non-blocking is specified, control returns from the function without waiting for IOP-side processing to end. Use sceSdCallBack() to set the end callback function to find out when IOP-side processing has completed.

If the next command is sent before IOP processing ends, an invalid operation may occur.

Non-blocking is currently not supported. Note that when non-blocking is specified, a -1 return value is returned and no processing is performed.

Return Value

0 if termination is normal. -1 in case of an error.

Chapter 6: Standard Kit Library/Sound System
Table of Contents

Functions **6-3**

sceSkInit 6-3
sceSkSsAllNoteOff 6-4
sceSkSsAllSoundOff 6-5
sceSkSsBind(SQ/MIDI/SONG/SESQ) 6-6
sceSkSsBindHDBD 6-8
sceSkSsBindTrack(SQ/MIDI/SONG/SESQ) 6-9
sceSkSsBindTrackHDBD 6-10
sceSkSsInit 6-11
sceSkSsPause(MIDI/SONG) 6-12
sceSkSsPlay(MIDI/SONG) 6-13
sceSkSsPlaySESQ 6-14
sceSkSsSend 6-15
sceSkSsSetDigitalOut 6-16
sceSkSsSetEffect 6-17
sceSkSsSetEffectMode 6-18
sceSkSsSetEffectVolume 6-19
sceSkSsSetMasterVolume 6-20
sceSkSsSetPlayAbsoluteTempo(MIDI/SONG) 6-21
sceSkSsSetPlayRelativeTempo(MIDI/SONG) 6-22
sceSkSsSetPlayTempo(MIDI/SONG) 6-23
sceSkSsSetPlayVolume(MIDI/SONG) 6-24
sceSkSsStatus(MIDI/SONG) 6-25
sceSkSsStop(MIDI/SONG) 6-26
sceSkSsStopSESQ 6-27
sceSkSsUnbind(SQ/MIDI/SONG/SESQ) 6-28
sceSkSsUnbindHDBD 6-29
sceSkSsUnbindTrack(SQ/MIDI/SONG/SESQ) 6-30
sceSkSsUnbindTrackHDBD 6-31
sceSkSsVoiceNoteOff 6-32
sceSkSsVoiceNoteOn 6-33
sceSkSsVoiceSetExpression 6-34
sceSkSsVoiceSetPanpot 6-35
sceSkSsVoiceSetPitchBend 6-36

Functions

sceSkInit

Initialize libsk environment

Library	*Introduced*	*Documentation last modified*
libsk	2.4	January 4, 2002

Syntax

int sceSkInit (	
unsigned int *option***)**	Initialization option (currently unused)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

This function initializes the libsk environment. In the current implementation, it returns 0 without performing any processing.

Return value

Always 0

sceSkSsAllNoteOff

Stop all sound generation for track

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsAllNoteOff (	
unsigned int *track_id***)**	Track id

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function stops all voices that are generating sound for the track specified by *track_id*.

Return value

0	Normal termination
< 0	Error

sceSkSsAllSoundOff

Forcibly stops all sound generation for track

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsAllSoundOff (

unsigned int *track_id***)**	Track id

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function forcibly stops all voices that are generating sound for the track specified by *track_id*. When this processing is performed, the release rates (RR) of all sound generation envelopes are set to the minimum.

Return value

0	Normal termination
< 0	Error

sceSkSsBind(SQ/MIDI/SONG/SESQ)

Register SQ data

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsBindSQ (	
*type***,**	Type of chunk to be registered in SQ data MIDI ... MIDI chunk SONG ... SONG chunk SESQ ... SE sequence chunk
unsigned int *id***,**	SQ id
unsigned int *sq_addr***,**	Starting address of area in IOP memory where SQ data is located
unsigned int *sq_size***)**	Size of SQ data
int sceSkSsBindMIDI (	
unsigned int *id***,**	SQ id
unsigned int *sq_addr***,**	Starting address of area in IOP memory where SQ data is located
unsigned int *sq_size***)**	Size of SQ data
int sceSkSsBindSONG (	
unsigned int *id***,**	SQ id
unsigned int *sq_addr***,**	Starting address of area in IOP memory where SQ data is located
unsigned int *sq_size***)**	Size of SQ data
int sceSkSsBindSESQ (	
unsigned int *id***,**	SQ id
unsigned int *sq_addr***,**	Starting address of area in IOP memory where SQ data is located
unsigned int *sq_size***)**	Size of SQ data

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function registers SQ data and obtains its SQ id. *type* indicates the chunk type to be used in the SQ data. During compilation, sceSkSsBindSQ() is replaced by either sceSkSsBindMIDI(), sceSkSsBindSONG(), or sceSkSsBindSESQ() according to the specified type. However, these functions can also be called directly.

When *id* is set to SCESK_AUTOASSIGNMENT, a free SQ id will be found and automatically assigned. The SQ id can also be assigned by directly specifying a numeric value for *id*. In that case, *id* must be in the following range.

- When *type* = MIDI or SONG:
  0 <= *id* < (maxentry value of skmidi module)

- When *type* = SESQ:
  0 <= *id* < (maxentry value of sksesq module)

If the specified *id* is not in the range shown above, SCESK_EINDEX_EXCEEDED is returned and processing will be abnormally terminated.

sq_addr should be set to the starting address of the area in IOP memory where the SQ data is located. *sq_size* should be set to the size of the SQ data.

Return value

>= 0	Valid SQ id
SCESK_EINDEX_EXCEEDED	Specified SQ id is out of range

sceSkSsBindHDBD

Register HD/BD data

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsBindHDBD (

unsigned int *id***,**	HD/BD id
unsigned int *hd_addr***,**	Starting address of area in IOP memory where HD data is located
unsigned int *hd_size***,**	HD data size
unsigned int *bd_addr***,**	Starting address of area in SPU2 local memory where BD data is located
unsigned int *bd_size***)**	BD data size

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function registers HD/BD data and obtains its HD/BD *id*. When *id* is set to SCESK_AUTOASSIGNMENT, a free HD/BD id will be found and automatically assigned. The HD/BD id can also be assigned by directly specifying a numeric value for *id*. In that case, *id* must be in the following range.

0 <= *id* < (maxtrack value of sksound module)

If the specified *id* is not in the range shown above, SCESK_EINDEX_EXCEEDED is returned and processing will be abnormally terminated.

hd_addr should be set to the starting address of the area in IOP memory where the HD data is located. *hd_size* should be set to the size of the HD data. BD data must be transferred to SPU2 local memory in advance. *bd_addr* should be set to the starting address of the area in IOP memory where the BD data is located. *bd_size* should be set to the size of the BD data.

Return value

>= 0	Valid HD/BD id
SCESK_EINDEX_EXCEEDED	Specified HD/BD id is out of range

sceSkSsBindTrack(SQ/MIDI/SONG/SESQ)

Register SQ id for track

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsBindTrackSQ (	
type,	Type of SQ data chunk that was specified when id was registered MIDI ... MIDI chunk SONG ... SONG chunk SESQ ... SE sequence chunk
unsigned int *track_id*,	Track id
unsigned int *sq_id*)	SQ id
int sceSkSsBindTrackMIDI (	
unsigned int *track_id*,	Track id
unsigned int *sq_id*)	SQ id
int sceSkSsBindTrackSONG (	
unsigned int *track_id*,	Track id
unsigned int *sq_id*)	SQ id
int sceSkSsBindTrackSESQ (	
unsigned int *track_id*,	Track id
unsigned int *sq_id*)	SQ id

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function registers an SQ id in a track and obtains its track id. *type* indicates the chunk type that was specified when *sq_id* was registered. During compilation, sceSkSsBindTrackSQ() is replaced by either sceSkSsBindTrackMIDI(), sceSkSsBindTrackSONG(), or sceSkSsBindTrackSESQ() according to the specified type. However, these functions can also be called directly.

When *track_id* is set to SCESK_AUTOASSIGNMENT, a free track id will be found and automatically assigned. The track id can also be assigned by directly specifying a numeric value for *track_id*. In that case, *track_id* must be in the following range.

0 <= *track_id* < (maxtrack value of sksound module)

If the specified *track_id* is not in the range shown above, SCESK_EINDEX_EXCEEDED is returned and processing will be abnormally terminated. *sq_id* should be set to an SQ id that was previously registered. If the specified *sq_id* has not been registered for the specified type, SCESK_ENOT_BOUND is returned and processing will be abnormally terminated.

Return value

>= 0	Valid track id
SCESK_EINDEX_EXCEEDED	Specified track id is out of range
SCESK_ENOT_BOUND	Specified SQ id is not registered

sceSkSsBindTrackHDBD

Register HD/BD id for track

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsBindTrackHDBD (	
unsigned int *track_id***,**	Track id
unsigned int *hdbd_id***,**	HD/BD id
unsigned int *bank_no***)**	Bank number (0 to 15)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function registers an HD/BD id in a track and obtains its track id. When *track_id* is set to SCESK_AUTOASSIGNMENT, a free track id will be found and automatically assigned. The track id can also be assigned by directly specifying a numeric value for *track_id*. In that case, *track_id* must be in the following range.

0 <= *track_id* < (maxtrack value of sksound module)

If the specified *track_id* is not in the range shown above, SCESK_EINDEX_EXCEEDED is returned and processing will be abnormally terminated. *hdbd_id* should be set to an HD/BD id that was previously registered. If the specified *hdbd_id* has not been registered, SCESK_ENOT_BOUND is returned and processing is abnormally terminated.

bank_no is set to the bank number where the HD/BD data specified by the HD/BD id, is registered. This HD/BD data can be used by specifying this bank number for the bank select MSB in score data (MIDI message).

The same *track_id* can be specified and the *hdbd_id* of the individual HD/BD data to be used for the required *bank_number* can also be specified.

```
/* Automatically assigning the track id and specifying bank 0 */

track = sceSkSsBindTrackHDBD (SCESK_AUTOASSIGNMENT, hdbd_id_0, 0);

/* Specifiying Æ hdbd_id_1 for bank 1 of this track */

return_val = sceSkSsBindTrackHDBD (track, hdbd_id_1, 1);
```

Return value

>= 0	Valid track id
SCESK_EINDEX_EXCEEDED	Specified track id is out of range
SCESK_ENOT_BOUND	Specified HD/BD id is not registered
SCESK_EINVALID_ARGUMENT	Specified bank number is out of range

sceSkSsInit

Initialize standard kit/sound system

Library	*Introduced*	*Documentation last modified*
libsk	2.4	January 4, 2002

Syntax

int sceSkSsInit (	
unsigned int *cb_priority***)**	libsdr callback thread priority

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function initializes the standard kit/sound system. It also initializes the internal libsd remote environment (libsdr). When sceSdRemoteInit() is called, *cb_priority* is passed directly as an argument of sceSdRemoteCallbackInit() and a callback thread is created. As a result, the user program need not call these functions.

Return value

>= 0	ID of EE thread that was activated for libsdr callback
< 0	Error

sceSkSsPause(MIDI/SONG)

Pause track performance (specified track of MIDI/SONG data)

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsPauseMIDI (	
unsigned int *track_id***)**	Track id
int sceSkSsPauseSONG (	
unsigned int *track_id***)**	Track id

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function pauses the performance of the data specified by the track id. To cancel the pause state, use the sceSkSsPlay(MIDI/SONG) function.

Return value

0	Normal termination
SCESK_EINDEX_EXCEEDED	Specified track id is out of range
SCESK_ENOT_BOUND	Specified SQ id is not registered
SCESK_EINVALID_ARGUMENT	Track id has not been registered by specified data
SCESK_EINVALID_STATUS	Specified track is not being performed

sceSkSsPlay(MIDI/SONG)

Play track (specified track of MIDI/SONG data)

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsPlayMIDI (	
unsigned int *track_id*,	Track id
unsigned int *no*)	MIDI block/SONG table number to be performed
int sceSkSsPlaySONG (	
unsigned int *track_id*,	Track id
unsigned int *no*)	MIDI block/SONG table number to be performed

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function plays the data specified by the track id. The MIDI block number (MIDI data specification) or SONG block number (SONG data specification) to be performed within the data is specified in *no*.

Return value

0	Normal termination
SCESK_EINDEX_EXCEEDED	Specified track id is out of range
SCESK_ENOT_BOUND	Specified SQ id is not registered
SCESK_EINVALID_ARGUMENT	Track id has not been registered by specified data Specified performance block/table number is out of range
SCESK_EINVALID_STATUS	Specified track is being performed

sceSkSsPlaySESQ

Play track (specified track of SE sequence data)

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsPlaySESQ (	
unsigned int *track_id***,**	Track id
unsigned int *set_no***,**	SE sequence set number to be played
unsigned int *seq_no***)**	SE sequence number within SE sequence set to be played

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function plays the data specified by the track id.

The SE sequence set number and SE sequence number to be performed within the data is specified in *set_no* and *seq_no*, respectively.

With the current implementation, the SE sequence ID is automatically released (unselected) after the SE sequence performance ends.

More than one performance start can be specified for a given track. At most 32 SE sequences can be performed within one track. If 32 sequences have already been performed, SCESK_ENOT_START is returned and processing will be abnormally terminated.

Return value

>= 0	SE sequence ID
SCESK_EINDEX_EXCEEDED	Specified track id is out of range
SCESK_ENOT_BOUND	Specified SQ id is not registered
SCESK_EINVALID_ARGUMENT	Track id is not registered by specified data or specified SE sequence set number/SE sequence number is out of range
SCESK_ENOT_START	SE sequence had already performed 32 sequences and an additional performance could not be done

sceSkSsSend

Transfer data between IOP memory and SPU2 local memory

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsSend (

unsigned int *mode***,**	Transfer direction Currently only SCESK_SEND_IOP2SPU can be specified
unsigned int *dma_ch***,**	SPU2 DMA channel to be used for the transfer
unsigned int *iop_addr***,**	Starting address of area in IOP memory
unsigned int *spu2_addr***,**	Starting address of area in SPU2 local memory
unsigned int *size***)**	Transfer size

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function transfers data between IOP memory and SPU2 local memory. Currently, only *mode* = SCESK_SEND_IOP2SPU is supported (IOP memory => SPU2 local memory transfer).

The starting address of the IOP memory area for the transfer is specified in *iop_addr*, and the starting address of the SPU2 local memory area for the transfer is specified in *spu2_addr*. *size* is set to the amount of data to be transferred.

The data is DMA transferred between IOP memory and SPU2 local memory. The SPU2 DMA channel to be used is specified in *dma_ch*.

Return value

>= 0	Actual transfer size
< 0	Transfer error

sceSkSsSetDigitalOut

Set optical digital output

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsSetDigitalOut (

unsigned int *mode***)**	Optical digital output mode	
	SCESK_DOUT_CD_NORMAL	CD media with no copy guard
	SCESK_DOUT_CD_COPY_PROHIBIT	CD media with copy guard
	SCESK_DOUT_DVD_NORMAL	DVD media with no copy guard
	SCESK_DOUT_DVD_COPY_PROHIBIT	DVD media with copy guard

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function sets the mode for optical digital output, which is one of the sound outputs.

Return value

Always 0

sceSkSsSetEffect

Enable/disable effect processing

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsSetEffect (

unsigned int *on_off***)**	Effect processing	
	SCESK_EFFECT_ON	Enable
	SCESK_EFFECT_OFF	Disable

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function sets whether to enable or disable effect processing for sound output.

Return value

Always 0

sceSkSsSetEffectMode

Set effect mode

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsSetEffectMode (	
unsigned int *mode***,**	Effect mode
unsigned int *delay***,**	Effect/display parameter (0 to 127)
unsigned int *feedback***)**	Effect/feedback parameter (0 to 127)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function sets the effect mode and parameters.

Table 6-1 *<mode>*

Specification Value	Mode
SCESK_EFFECT_MODE_OFF	off
SCESK_EFFECT_MODE_ROOM	Room
SCESK_EFFECT_MODE_STUDIO_A	Studio (small)
SCESK_EFFECT_MODE_STUDIO_B	Studio (medium)
SCESK_EFFECT_MODE_STUDIO_C	Studio (large)
SCESK_EFFECT_MODE_HALL	Hall
SCESK_EFFECT_MODE_SPACE	Space echo
SCESK_EFFECT_MODE_ECHO	Echo
SCESK_EFFECT_MODE_DELAY	Delay
SCESK_EFFECT_MODE_PIPE	Pipe echo

If SCESK_EFFECT_MODE_CLEAR_WA is ORed in with the *mode* specification, the effect work area will be cleared when the mode is set. DMA channel 0 is used for clearing the effect work area. Internally, processing is synchronized with the end of the DMA transfer that clears the effect work area, so the user program need not be concerned with DMA transfer termination processing.

<delay>

This argument is valid only when *mode* is ECHO or DELAY. If any other mode is specified, *delay* should be set to 0. Delay time should be set in the range 0 to 127.

<feedback>

This argument is valid only when *mode* is ECHO or DELAY. If any other mode is specified, feedback should be set to 0. *feedback* should be set in the range 0 to 127.

Return value

Always 0

sceSkSsSetEffectVolume

Set effect volume

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsSetEffectVolume (

unsigned int *l***,**	Effect volume/L channel (0 to 127)
unsigned int *r***)**	Effect volume/R channel (0 to 127)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function sets the values of the effect volume for sound output.

Return value

Always 0

sceSkSsSetMasterVolume

Set master volume

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsSetMasterVolume (	
unsigned int *l***,**	Master volume/L channel (0 to 127)
unsigned int *r***)**	Master volume/R channel (0 to 127)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function sets the values of the master volume for sound output.

Return value

Always 0

sceSkSsSetPlayAbsoluteTempo(MIDI/SONG)

Set absolute tempo for track play (specified track of MIDI/SONG data)

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsSetPlayAbsoluteTempoMIDI (	
unsigned int *track_id*,	Track id
unsigned int *tempo*)	Tempo (20 to 255)
int sceSkSsSetPlayAbsoluteTempoSONG (	
unsigned int *track_id*,	Track id
unsigned int *tempo*)	Tempo (20 to 255)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function sets the tempo to be used when the data specified by the track id is played. *tempo* is set to the number of quarter notes per minute to be played.

Return value

0	Normal termination
SCESK_EINDEX_EXCEEDED	Specified track id is out of range
SCESK_ENOT_BOUND	Specified SQ id is not registered
SCESK_EINVALID_ARGUMENT	Track id is not registered by specified data or specified value is out of range

sceSkSsSetPlayRelativeTempo(MIDI/SONG)

Set relative tempo for track play (specified track of MIDI/SONG data)

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsSetPlayRelativeTempoMIDI (	
unsigned int *track_id***,**	Track id
unsigned int *tempo***)**	Tempo (relative value where 0x100 means 100%)
int sceSkSsSetPlayRelativeTempoSONG (	
unsigned int *track_id***,**	Track id
unsigned int *tempo***)**	Tempo (relative value where 0x100 means 100%)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function sets the tempo to be used when the data specified by the track id is played. *tempo* specifies the relative value from the current tempo, where 0x100 means 100%.

Return value

0	Normal termination
SCESK_EINDEX_EXCEEDED	Specified track id is out of range
SCESK_ENOT_BOUND	Specified SQ id is not registered
SCESK_EINVALID_ARGUMENT	Track id is not registered by specified data or specified value is out of range

sceSkSsSetPlayTempo(MIDI/SONG)

Set tempo for track play (specified track of MIDI/SONG data)

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsSetPlayTempoMIDI (

type,	Type of tempo to be specified ABSOLUTE ... Absolute tempo specification RELATIVE ... Relative tempo specification
unsigned int *track_id*,	Track id
unsigned int *tempo*)	Tempo (Absolute: 20 to 255; Relative: Relative value, where 0x100 is set for 100%)

int sceSkSsSetPlayTempoSONG (

type,	Type of tempo to be specified ABSOLUTE ... Absolute tempo specification RELATIVE ... Relative tempo specification
unsigned int *track_id*,	Track id
unsigned int *tempo*)	Tempo (Absolute: 20 to 255; Relative: Relative value, where 0x100 is set for 100%)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function sets the tempo to be used when the data specified by the track id is played. During compilation, sceSkSsSetPlayTempoMIDI() is replaced by either sceSkSsSetPlayAbsoluteTempoMIDI() or sceSkSsSetPlayRelativeTempoMIDI() and sceSkSsSetPlayTempoSONG() is replaced by either sceSkSsSetPlayAbsoluteTempoSONG() or sceSkSsSetPlayRelativeTempoSONG() according to the specified type. However, these functions can also be called directly.

Relative tempo specifies the relative value from the current tempo, where 0x100 means 100%. Absolute tempo specifies the number of quarter notes per minute to be played.

Return value

0	Normal termination
SCESK_EINDEX_EXCEEDED	Specified track id is out of range
SCESK_ENOT_BOUND	Specified SQ id is not registered
SCESK_EINVALID_ARGUMENT	Track id is not registered by specified data or specified value is out of range

sceSkSsSetPlayVolume(MIDI/SONG)

Specify play volume of track (specified track of MIDI/SONG data)

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsSetPlayVolumeMIDI (	
unsigned int *track_id***,**	Track id
unsigned int *volume***)**	Volume (0 to 127)
int sceSkSsSetPlayVolumeSONG (	
unsigned int *track_id***,**	Track id
unsigned int *volume***)**	Volume (0 to 127)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function sets the volume to be used when the data specified by the track id is played.

Return value

0	Normal termination
SCESK_EINDEX_EXCEEDED	Specified track id is out of range
SCESK_ENOT_BOUND	Specified SQ id is not registered
SCESK_EINVALID_ARGUMENT	Track id is not registered by specified data

sceSkSsStatus(MIDI/SONG)

Get performance state (specified track of MIDI/SONG data)

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsStatusMIDI (	
unsigned int *track_id***,**	Track id
unsigned int *command***)**	State acquisition command SCESK_ISPLAYING ... Is that track being performed? SCESK_ISDATAEND ... Has that track reached end of performance? SCESK_POSITION ... Current position of that track's performance (units: ticks)
int sceSkSsStatusSONG (	
unsigned int *track_id***,**	Track id
unsigned int *command***)**	State acquisition command SCESK_ISPLAYING ... Is that track being performed? SCESK_ISDATAEND ... Has that track reached end of performance? SCESK_POSITION ... Current position of that track's performance (units: ticks)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function gets the state of the performance for the data that is being performed by the track specified by the track id.

Return value

command	=	SCESK_ISPLAYING:
	> 0	Performance is in progress
	0	Performance is stopped
command	=	SCESK_ISDATAEND:
	> 0	Reached end
	0	Has not reached end
command	=	SCESK_POSITION:
	>= 0	Performance position (units: ticks)

sceSkSsStop(MIDI/SONG)

Stop track performance (specified track of MIDI/SONG data)

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsStopMIDI (	
unsigned int *track_id***)**	Track id
int sceSkSsStopSONG (	
unsigned int *track_id***)**	Track id

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function stops the performance of the data specified by the track id and moves the performance position to the beginning of the tune.

If a stop operation is performed for a track that is already paused, only the processing for moving the performance target to the beginning of the tune is performed.

Return value

0	Normal termination
SCESK_EINDEX_EXCEEDED	Specified track id is out of range
SCESK_ENOT_BOUND	Specified SQ id is not registered
SCESK_EINVALID_ARGUMENT	Track id is not registered by specified data

sceSkSsStopSESQ

Stop track performance (specified track of SE sequence data)

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsStopSESQ (	
unsigned int *track_id***,**	Track id
unsigned int *sesq_id***)**	SE sequence ID that was returned when performance started

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function stops the performance of the data specified by the track id.

In the current implementation, the sequence ID is automatically released (unselected) after the performance of the SE sequence has ended, so it is possible that the SE sequence ID may already be released when sceSkSsStopSESQ() attempts to stop the performance. In that case, SCESK_ENOT_STOP is returned.

Return value

0	Normal termination
SCESK_EINDEX_EXCEEDED	Specified track id is out of range
SCESK_ENOT_BOUND	Specified SQ id is not registered
SCESK_EINVALID_ARGUMENT	Track id is not registered by specified data
SCESK_ENOT_STOP	Termination processing could not be performed for specified SE sequence ID

sceSkSsUnbind(SQ/MIDI/SONG/SESQ)

Cancel SQ data registration

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsUnbindSQ (	
*type***,**	SQ data chunk type specified when *id* was registered MIDI ... MIDI chunk SONG ... SONG chunk SESQ ... SE sequence chunk
unsigned int *id***)**	SQ id
int sceSkSsUnbindMIDI (	
unsigned int *id***)**	SQ id
int sceSkSsUnbindSONG (	
unsigned int *id***)**	SQ id
int sceSkSsUnbindSESQ (	
unsigned int *id***)**	SQ id

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function cancels the registration of an SQ id that was previously registered.

The SQ data chunk type that was specified when the SQ id was registered, is specified for *type*. During compilation, sceSkSsUnbindSQ() is replaced by either sceSkSsUnbindMIDI(), sceSkSsUnbindSONG(), or sceSkSsUnbindSESQ() according to the specified type. However, these functions can also be called directly.

Cancelling a registration only changes the state in the standard kit/sound system and does not affect the IOP memory area that was specified during registration. If this area is no longer needed, it should be freed after this function is called.

Return value

0	Normal termination
SCESK_EINDEX_EXCEEDED	Specified SQ id is out of range

sceSkSsUnbindHDBD

Cancel HD/BD data registration

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsUnbindHDBD (	
unsigned int *id***)**	HD/BD id

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function cancels the registration of an HD/BD id that was previously registered.

Cancelling a registration only changes the state in the standard kit/sound system and does not affect either the IOP memory area or SPU2 local memory area that was specified during registration. If these areas are no longer needed, they should be freed after this function is called.

Return value

0	Normal termination
SCESK_EINDEX_EXCEEDED	Specified HD/BD id is out of range

sceSkSsUnbindTrack(SQ/MIDI/SONG/SESQ)

Cancel track registration

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsUnbindTrackSQ (	
type,	SQ data chunk type specified when *id* was registered MIDI ... MIDI chunk SONG ... SONG chunk SESQ ... SE sequence chunk
int sceSkSsUnbindTrackMIDI (	
unsigned int *track_id***)**	Track id
int sceSkSsUnbindTrackSONG (	
unsigned int *track_id***)**	Track id
int sceSkSsUnbindTrackSESQ (	
unsigned int *track_id***)**	Track id

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function cancels the registration of a track id that was previously registered.

During compilation, sceSkSsUnbindTrackSQ() is replaced by either sceSkSsUnbindTrackMIDI(), sceSkSsUnbindTrackSONG(), or sceSkSsUnbindTrackSESQ() according to the specified type. However, these functions can also be called directly.

Cancelling a registration only changes the state in the standard kit/sound system and does not affect the SQ data with an SQ id that was registered with the specified track id. If this SQ data is no longer needed, its registration should be individually cancelled after this function is called.

Return value

0	Normal termination
SCESK_EINDEX_EXCEEDED	Specified track id is out of range
SCESK_ENOT_BOUND	Specified track id is not registered

sceSkSsUnbindTrackHDBD

Cancel track registration

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsUnbindTrackHDBD (	
unsigned int *track_id***)**	Track id

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function cancels the registration of a track id that was previously registered.

Cancelling a registration only changes the state in the standard kit/sound system and does not affect the HD/BD data with an HD/BD id that was registered with the specified track id. If this HD/BD data is no longer needed, its registration should be individually cancelled after this function is called.

Return value

0	Normal termination
SCESK_EINDEX_EXCEEDED	Specified track id is out of range
SCESK_ENOT_BOUND	Specified track id is not registered

sceSkSsVoiceNoteOff

Mute one-shot sound generation

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsVoiceNoteOff (

unsigned int *track_id***,**	Track id
unsigned int *v_id***,**	Voice id (0 to 126)
unsigned int *bank_no***,**	Bank number
unsigned int *program_no***,**	Program number (HD attribute)
unsigned int *midi_ch***,**	MIDI ch
unsigned int *note***)**	Note number (0 to 127)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function mutes the voice for which one-shot sound generation was performed for the track specified by the track id. An extended MIDI message is generated internally. *v_id* should be set to the voice ID that was specified when the sound was generated. *bank_no* should be set to the bank number that was specified when the sound was generated. *program_no* should be set to the program number that was specified when the sound was generated. *midi_ch* should be set to the *midi_ch* that was specified when the sound was generated. *note* should be set to the pitch to be muted.

Return value

Always 0

sceSkSsVoiceNoteOn

Generate one-shot sound

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsVoiceNoteOn (

unsigned int *track_id*,	Track id
unsigned int *v_id*,	Voice id (0 to 126)
unsigned int *bank_no*,	Bank number
unsigned int *program_no*,	Program number (HD attribute)
unsigned int *midi_ch*,	MIDI ch
unsigned int *note*,	Note number (0 to 127)
unsigned int *velocity*,	Generated sound volume (0 to 127)
unsigned int *panpot***)**	Panpot (0/L to 64/CENTER to 127/R)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function generates a single sound using the HD/BD data with a track specified by the track id.

An extended MIDI message is generated internally.

The voice ID for distinguishing the generated sound is specified for *v_id*. The bank number that was specified with sceSkSsBindTrackHDBD() for the track corresponding to *track_id* is specified for *bank_no*. The program number that is included in the HD attribute is specified for *program_no*. *midi_ch* is used to distinguish the tone. Sixteen tones can be distinguished with a *program_no* specification for each track (equivalent to one tone per MIDI channel, for a total of 16 tones that can be simultaneously specified). The pitch, volume, and panpot to be generated are specified for *note*, *velocity*, and *panpot*, respectively.

Return value

Always 0

sceSkSsVoiceSetExpression

Set expression for one-shot generated voice

Library	*Introduced*	*Documentation last modified*
libsk	2.4.2	December 3, 2001

Syntax

int sceSkSsVoiceSetExpression (

unsigned int *track_id***,**	Track id
unsigned int *v_id***,**	Voice id (0-126)
unsigned int *midi_ch***,**	MIDI ch
unsigned int *note***,**	Note number (0-127)
unsigned int *expression***)**	Expression (0-127)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function sets an expression for the voice which was generated with a one-shot sound in the track specified by the track id. An extended MIDI message is generated internally.

v_id should be set to the voice ID that was specified when the sound was generated. *midi_ch* should be set to the *midi_ch* that was specified when the sound was generated. *note* should be set to the musical interval to be muted. *expression* should be set to the expression.

Return value

Always 0

sceSkSsVoiceSetPanpot

Change panpot position of one-shot generated voice

Library	*Introduced*	*Documentation last modified*
libsk	2.4	October 1, 2001

Syntax

int sceSkSsVoiceSetPanpot (	
unsigned int *track_id***,**	Track id
unsigned int *v_id***,**	Voice id (0 to 126)
unsigned int *midi_ch***,**	MIDI ch
unsigned int *note***,**	Note number (0 to 127)
unsigned int *panpot***)**	Panpot (0/L to 64/CENTER to 127/R)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function changes the panpot position for the voice for which one-shot sound generation was performed for the track specified by the track id. An extended MIDI message is generated internally.

v_id should be set to the voice ID that was specified when the sound was generated. *midi_ch* should be set to the *midi_ch* that was specified when the sound was generated. *note* should be set to the pitch to be muted. *panpot* should be set to the desired panpot position.

Return value

Always 0

sceSkSsVoiceSetPitchBend

Add pitchbend effect to one-shot generated voice

Library	*Introduced*	*Documentation last modified*
libsk	2.4.2	December 3, 2001

Syntax

int sceSkSsVoiceSetPitchBend (

unsigned int *track_id*,	Track id
unsigned int *v_id*,	Voice id (0-126)
unsigned int *midi_ch*,	MIDI ch
unsigned int *note*,	Note number (0-127)
unsigned int *pitchbend*)	Pitchbend (0(maximum negative effect) to 0x2000(no effect) to 0x3fff(maximum positive effect))

Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

Description

This function adds a pitchbend effect to the voice which was generated with a one-shot sound in the track specified by the track id. An extended MIDI message is generated internally.

v_id should be set to the voice ID that was specified when the sound was generated. *midi_ch* should be set to the *midi_ch* that was specified when the sound was generated. *note* should be set to the musical interval to be muted. *pitchbend* is set to the pitchbend to be added. Pitchbend values 00 00 to 40 00 to 7f 7f, which are represented in MIDI by 7 bits each for MSB/LSB, are set in a contiguous bit array and converted to the numeric values 0x0000 to 0x2000 to 0x3fff.

Return value

Always 0

Chapter 7: SPU2 Local Memory Management Library
Table of Contents

Functions **7-3**

sceSpu2MemAllocate 7-3

sceSpu2MemFree 7-4

sceSpu2MemInit 7-5

sceSpu2MemQuit 7-6

Functions

sceSpu2MemAllocate

Allocate SPU2 local memory

Library	*Introduced*	*Documentation last modified*
libspu2m	2.4	January 4, 2002

Syntax

int sceSpu2MemAllocate (	
unsigned int *req_size***)**	Requested size of area (bytes)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

This function allocates an area in SPU2 local memory with a size specified by *req_size*. Due to SPU2 limitations, if *req_size* is not a multiple of 16, it is rounded up so that it is a multiple of 16 bytes.

The *size* argument of sceSpu2MemInit() will be used as an upper limit on the number of area allocations, that is, the number of pieces into which SPU2 local memory can be subdivided. If that limit is exceeded when sceSpu2MemAllocate is called, an error will be returned even if an unallocated area having a size of *req_size* remains.

Return value

>= 0	Starting address of allocated area
<0	Error

sceSpu2MemFree

Free SPU2 local memory

Library	*Introduced*	*Documentation last modified*
libspu2m	2.4	January 4, 2002

Syntax

void sceSpu2MemFree (	
unsigned int *addr***)**	Starting address of area in SPU2 local memory

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

This function frees an area in SPU2 local memory, which was previously allocated using sceSpu2MemAllocate().

Return value

None

sceSpu2MemInit

Initialize libspu2m environment

Library	*Introduced*	*Documentation last modified*
libspu2m	2.4	January 4, 2002

Syntax

int sceSpu2MemInit (

void **table*,	Starting address of memory management table
unsigned int *size*,	Size of memory management table (size divided by SCESPU2MEM_TABLE_UNITSIZE - 1)
unsigned int *eff_opt*)	Whether or not to reserve effect work area SCESPU2MEM_NO_EFFECT Do not reserve effect work area SCESPU2MEM_USE_EFFECT Reserve effect work area

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

This function initializes the libspu2m environment. The memory management table is specified according to table and size. The memory management table is a global variable that must have an area of (SCESPU2MEM_TABLE_UNITSIZE *(size + 1)) bytes. Area can be allocated in SPU2 local memory only by a multiple of the size argument.

The *eff_opt* argument can be used to specify whether or not an effect work area in SPU2 local memory should be considered. To use effects for a performance in the SPU2, be sure to use SCESPU2MEM_USE_EFFECT. SCESPU2MEM_NO_EFFECT can be specified if no effects will be used.

The size that is allocated for the user area in SPU2 local memory will vary as follows according to the specification of *eff_opt*.

eff_opt = SCESPU2MEM_USE_EFFECT
  1,814,512 (0x1baff0) bytes

eff_opt = SCESPU2MEM_NO_EFFECT
  2,076,656 (0x1faff0) bytes

To enable 32 areas to be allocated and to use effects, the following should be specified.

```
/* Allocation of 32 areas */

static unsigned char tab[SCESPU2MEM_TABLE_UNITSIZE * (32 + 1)];

sceSpu2MemInit (tab, 32,             /* Number of areas: 32 */

                    SCESPU2MEM_USE_EFFECT); /* Effects are used */
```

Return value

0	Normal termination
<0	Error

sceSpu2MemQuit

Terminate libspu2m environment

Library	*Introduced*	*Documentation last modified*
libspu2m	2.4	January 4, 2002

Syntax

int sceSpu2MemQuit (void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

This function terminates the libspu2m environment.

Return value

Always 0