PlayStation®2 IOP Library Reference
Release 2.4

Sound Libraries

Publication date: October 2001

Sony Computer Entertainment Inc.
1-1, Akasaka 7-chome, Minato-ku
Tokyo 107-0052, Japan

Sony Computer Entertainment America
919 E. Hillsdale Blvd.
Foster City, CA 94404, U.S.A.

Sony Computer Entertainment Europe
30 Golden Square
London W1F 9LD, U.K.

Summary Table of Contents

About This Manual

This is the Runtime Library Release 2.4 version of the *PlayStation®2 IOP Library Reference - Sound Libraries* manual.

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

Changes Since Last Release

Chapter 3: CSL Hardware Synthesizer (modhsyn)

- In the "Description" sections of the following functions, a description about multithread environment has been added.

  sceHSyn_ MSGetVoiceEnvelopeByID()
  sceHSyn_MSGetVoiceStateByID()
  sceHSyn_AllNoteOff()
  sceHSyn_AllSoundOff()
  sceHSyn_ATick()
  sceHSyn_GetChStat()
  sceHSyn_ResetAllControler()
  sceHSyn_SEAllNoteOff()
  sceHSyn_SEAllSoundOff()
  sceHSyn_SERetrieveVoiceNumberByID()
  sceHSyn_SESetMaxVoices()
  sceHSyn_SetEffectAttr()
  sceHSyn_SetReservVoice()
  sceHSyn_SetVoiceStatBuffer()
  sceHSyn_SetVolume()

- A description of the following function has been added.

  sceHSyn_GetReservVoice()

- The "Calling Condition" of the following function has been corrected.

  sceHSyn_Init()

Chapter 4: CSL MIDI Stream Generation (modmsin)

- The "Calling Condition" of sceMSIn_Init() has been corrected.
- In the "Description" section of the following functions, a description about multithread environment has been added.

  sceMSIn_NoteOff()
  sceMSIn_NoteOn()
  sceMSIn_NoteOnEx()
  sceMSIn_ProgramChange()
  sceMSIn_PutExcMsg()

sceMSIn_PutHsMsg()
sceMSIn_PutMsg()

Chapter 5: CSL SE Stream Generation (for IOP) (modsein)

- Descriptions of the following functions have been added.

  sceSEIn_MakeAllNoteOff()
  sceSEIn_MakeAllNoteOffMask()

- In the "Description" section of the following functions, a description about multithread environment has been added.

  sceSEIn_MakeAmpLFO()
  sceSEIn_MakeNoteOn()
  sceSEIn_MakePitchLFO()
  sceSEIn_MakePitchOn()
  sceSEIn_MakeTimePanpot()
  sceSEIn_MakeTimePitch()
  sceSEIn_MakeTimeVolume()
  sceSEIn_NoteOff()
  sceSEIn_NoteOn()
  sceSEIn_PitchOn()
  sceSEIn_PutMsg()
  sceSEIn_PutSEMsg()

Chapter 7: CSL MIDI Sequencer (modmidi)

- Descriptions of the following functions have been added.

  sceMidi_MidiGetUSecTempo()
  sceMidi_MidiSetUSecTempo()

- In the "Description" section of the following functions, a description about multithread environment has been added.

  sceMidi_ATick()
  sceMidi_MidiPlaySwitch()
  sceMidi_MidiSetAbsoluteTempo()
  sceMidi_MidiSetLocation()
  sceMidi_MidiSetRelativeTempo()
  sceMidi_MidiSetVolume()
  sceMidi_MidiVolumeChange()
  sceMidi_SelectMidi()
  sceMidi_SelectSong()
  sceMidi_SongPlaySwitch()
  sceMidi_SongSetAbsoluteTempo()
  sceMidi_SongSetLocation()
  sceMidi_SongSetRelativeTempo()
  sceMidi_SongSetVolume()

sceMidi_SongVolumeChange()

- The "Calling Conditions" of the following function have been corrected.

  sceMidi_Init()

- In the "Argument" section of sceMidi_MidiSetVolume(), the description about master volume specification has been corrected.

Chapter 9: Low-Level Sound Library

- Descriptions of the following functions have been added.

  sceSdGetTransIntrHandlerArgument()
  sceSdGetSpu2IntrHandlerArgument()

- In the "Description" section of the sceSdEffectAttr structure, the description of mode has been corrected.
- The "Calling Conditions" of the following functions have been corrected.

  sceSdBlockTrans()
  sceSdInit()
  sceSdSetAddr()
  sceSdSetCoreAttr()
  sceSdSetEffectAttr()
  sceSdSetParam()
  sceSdSetSpu2IntrHandler()
  sceSdSetSwitch()
  sceSdSetTransIntrHandler()
  sceSdVoiceTrans()

- In the "Description" section of sceSdClearEffectWorkArea(), a description of channel specification has been added.
- In the "Description" section of sceSdInit(), the description of interrupt controller has been corrected.
- In the "Description" section of the following functions, a description about multithread environment has been added.

  sceSdProcBatch()
  sceSdProcBatchEx()
  sceSdSetAddr()
  sceSdSetCoreAttr()
  sceSdSetEffectAttr()
  sceSdSetParam()
  sceSdSetSwitch()

- A description about function calls has been added to the following functions.

  sceSdSetSpu2IntrHandler()
  sceSdSetTransIntrHandler()

- In the "Description" section of sceSdVoiceTrans(), a description about transfer confirmation has been added.

Chapter 10: CSL SE Sequencer (modsesq)

- The "Calling Conditions" of sceSESq_Init() have been corrected.
- In the "Description" section of the following functions, a description about multithread environment has been added.

  sceSESq_ATick()
  sceSESq_SelectSeq()
  sceSESq_SeqPlaySwitch()
  sceSESq_SeqTerminateVoice()
  sceSESq_UnselectSeq()

Related Documentation

Library specifications for the EE can be found in the *PlayStation®2 EE Library Reference* manuals and the *PlayStation®2 EE 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: SPU2 Waveform Data Encoding Module
Table of Contents

Structures

sceSpuEncodeEnv

SPU2 waveform data encoding attributes

Library	*Introduced*	*Documentation last modified*
spucodec	2.2	March 23, 2001

Structure

typedef struct {

short **src;*	16-bit straight PCM data address
short **dest;*	PlayStation-original waveform data
short **work;*	Work area used when encoding
int *size;*	16-bit straight PCM data size (units: bytes)
int *loop_start;*	Loop starting point in PCM data (units: bytes)
int *loop;*	Loop waveform generation specification Specify SPUCODEC_ENCODE_LOOP or SPUCODEC_ENCODE_NO_LOOP
int *byte_swap;*	PCM data endian specification Specify SPUCODEC_ENCODE_ENDIAN_LITTLE or SPUCODEC_ENCODE_ENDIAN_BIG
int *proceed;*	Whole or divided encoding and progress specification Specify SPUCODEC_ENCODE_WHOLE, SPUCODEC_ENCODE_START, SPUCODEC_ENCODE_CONTINUE, or SPUCODEC_ENCODE_END
int *quality;*	Encode quality specification Specify SPUCODEC_ENCODE_MIDDLE_QUALITY or SPUCODEC_ENCODE_HIGH_QUALITY

} sceSpuEncodeEnv;

Description

This structure specifies the SPU2 waveform data encoding attributes to be used by sceSpuCodecEncode(). For details about how to use this structure, see sceSpuCodecEncode().

Return value

None

See also

sceSpuCodecEncode()

Functions

sceSpuCodecEncode

Encode waveform data

Library	*Introduced*	*Documentation last modified*
spucodec	2.2	March 23, 2001

Structure

int sceSpuCodecEncode (

sceSpuEncodeEnv **env***)**	SPU2 waveform data encoding attributes

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function encodes the 16-bit straight PCM data that is in the area specified by the SPU2 waveform data encoding attributes *env->src* and *env->size* and outputs the resulting SPU2 waveform data (equivalent to VAG but without header information) to the area that starts at the member *env->dest*. The return value of the function is set to the size after encoding.

The size of the 16-bit straight PCM data is specified in bytes for *env->size*.

To create a loop waveform, specify SPUCODEC_ENCODE_LOOP for *env->loop* and in *env->loop_start*, specify the loop starting point of the PCM data as a relative value in bytes from *env->src*. At this time, if *env->loop_start* is not a multiple of 56 (28 samples), the loop starting point is set at a location rounded down to a multiple of 56.

To create a non-loop waveform, specify SPUCODEC_ENCODE_NO_LOOP for *env->loop*. In this case, the *env->loop_start* specification is ignored.

Specify SPUCODEC_ENCODE_ENDIAN_BIG (16-bit big endian) or SPUCODEC_ENCODE_ENDIAN_LITTLE (16-bit little endian) for *env->byte_swap*, depending on the endian property of the PCM data.

Specify whole or divided encoding and the progress in *env->proceed*.

Table 1-1

env->proceed	Encoding specification
SPUCODEC_ENCODE_WHOLE	Whole encoding
SPUCODEC_ENCODE_START	Divided encoding start
SPUCODEC_ENCODE_CONTINUE	Divided encoding continuation
SPUCODEC_ENCODE_END	Divided encoding end

When *env->proceed* is not SPUCODEC_ENCODE_WHOLE, the area specified by *env->size* starting at *env->src* is encoded in an area-divided form in which the area that includes the starting block is encoded by SPUCODEC_ENCODE_START, intermediate areas are consecutively encoded by

SPUCODEC_ENCODE_CONTINUE, and the area that includes the final block is encoded by SPUCODEC_ENCODE_END.

At this time, if *env->size* is not a multiple of 56 (28 sample), encoding is performed in a form in which the end of the data is padded with zeros so that the size becomes a multiple of 56, and the continuity of the waveform data that is finally generated will be lost. Therefore, if you want to ensure that the waveform data is continuous, make sure that *env->size* is a multiple of 56 and perform divided encoding. Even when *env->proceed* is SPUCODEC_ENCODE_WHOLE, whole encoding is performed in a form in which the end of the data is padded with zeros so that *env->size* will be a multiple of 56.

To use a specific area as a work area during encoding, specify the starting address of that area in *env->work*. A 168-byte area starting at the specified address will be used. If NULL is specified for *env->work*, an automatic variable (=stack) is used internally. However, when *env->quality* is SPUCODEC_ENCODE_HIGH_QUALITY, only NULL can be specified. When quality versus speed are considered, setting *env->quality* to SPUCODEC_ENCODE_MIDDLE_QUALITY emphasizes speed over quality when encoding and setting *env->quality* to SPUCODEC_ENCODE_HIGH_QUALITY emphasizes quality over speed when encoding.

Return value

The size of the SPU2 waveform data that was created in env->dest by the encoding is returned.

If an error occurs, SPUCODEC_ENCODE_ERROR is returned.

Chapter 2: CSL MIDI Delay
Table of Contents

Structures

sceMidiDelay_DelayBuffer

Delay buffer

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

Structure

typedef struct {

unsigned int *delayBfSize;*	Delay buffer (data[]) byte count
unsigned int *rp, wp;*	rp: Delay buffer read pointer wp: Delay buffer write pointer
unsigned short *curTime;*	Current time
unsigned short *delayTime;*	Delay time (ATick() call frequency)
unsigned char delayBf[*0];*	Delay buffer Actually, this is delayBf[delayBfSize].

} sceMidiDelay_DelayBuffer;

Description

Structure for the delay buffer corresponding to the input buffer.

Functions

sceMidiDelay_ATick

Interrupt processing

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

Syntax

int sceMidiDelay_ATick(	
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-enabled state)

Description

Called from an interrupt at regular intervals.

Return value

If processing was successful: 0

sceMidiDelay_Flush

Output all delay buffer data

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

Syntax

int sceMidiDelay_Flush(	
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-enabled state)

Description

Outputs all delay buffer data to the output buffer regardless of the delay time.

Return value

If processing was successful: 0

sceMidiDelay_GetDelayBuffer

Get delay buffer address

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

Syntax

sceMidiDelay_DelayBuffer *sceMidiDelay_GetDelayBuffer(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number***)**	Input port number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Gets the delay buffer address corresponding to port_number.

Return value

Delay buffer address

sceMidiDelay_Init

Initialization

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

Syntax

int sceMidiDelay_Init(	
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-enabled state)

Description

Initializes the internal environment of the module.

Return value

If processing was successful: 0

Chapter 3: CSL Hardware Synthesizer
Table of Contents

Structures

sceHSyn_EffectAttr

Effect parameters

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

Structure

```
typedef struct {
    int core;
#define SCEHS_REV_MODE_OFF          0
#define SCEHS_REV_MODE_ROOM         1
#define SCEHS_REV_MODE_STUDIO_A     2
#define SCEHS_REV_MODE_STUDIO_B     3
#define SCEHS_REV_MODE_STUDIO_C     4
#define SCEHS_REV_MODE_HALL         5
#define SCEHS_REV_MODE_SPACE        6
#define SCEHS_REV_MODE_ECHO         7
#define SCEHS_REV_MODE_DELAY        8
#define SCEHS_REV_MODE_PIPE         9
#define SCEHS_REV_MODE_MAX          10
#define SCEHS_REV_MODE_CLEAR_WA     (1<<8)
     int mode;
     short depth_L, depth_R;
     int delay;
     int feedback;
     short vol_l, vol_r;                        Effect (depth) return volume
} sceHSyn_EffectAttr;
```

Note: Other members are the same as for libsd.h sceSdEffectAttr.

Description

Sets the effect attributes.

In the current implementation, the values of the effect (depth) return volume should be specified such that:

depth_L == vol_L, depth_R == vol_R.

sceHSyn_VoiceStat

Module state

Library	*Introduced*	*Documentation last modified*
modhysn	2.1	January 4, 2001

Structure

typedef struct {

int *pendingVoiceCount;*	Number of voices waiting to be generated (pending)
int *workVoiceCount;*	Number of voices being generated (including KEY_OFF)
unsigned char *voice_state* **[sceHSyn_NumCore] [sceHSyn_NumVoice];**	Voice state bit-7: Data enable bit When bit-7 == 1, the contents of bit-0 to bit-6 are valid bit-4,5,6: Voice state sceHSyn_VoiceStat_Free Free sceHSyn_VoiceStat_Pending Pending sceHSyn_VoiceStat_KeyOn Key on (being generated) sceHSyn_VoiceStat_KeyOff Key off (being generated) bit-0,1,2,3: Port number being used
unsigned short *voice_env* **[sceHSyn_NumCore] [sceHSyn_NumVoice];**	Envelope value (valid only when sceHSyn_VoiceStat_KeyOn or sceHSyn_VoiceStat_KeyOff)

} sceHSyn_VoiceStat;

Description

Gets the module state.

The following are provided as voice_state member handling macros:

sceHSyn_GetVoiceStat(voice_state[?][?] get voice state

sceHSyn_GetVoiceCtrlPort(voice_state[?][?]) get port used

For both, when bit-7 == 0, -1 is returned.

sceHSynChStat

Structure for getting the voice usage state for each channel

Library	*Introduced*	*Documentation last modified*
modhysn	2.1	January 4, 2001

Structure

typedef struct {

unsigned char *ch[16];*	The voice usage state of channel XX is entered for ch[XX] It can be examined with the following bits to determine the state: sceHSynChStat_KeyOn: KEY ON is set, and a voice is being generated sceHSynChStat_Hold: Although a MIDI Note Off Message was received, since HOLD ON is active, voice generation will continue sceHSynChStat_KeyOff: KEY OFF is set, and voice generation has not yet ended An empty (no voice is being generated) channel is a channel for which ch[XX] == 0

} sceHSynChStat;

Description

Receives the result of the check of voice generation state of each channel in sceHSyn_GetChStat().

sceHSynEnv

Input environment

Library	*Introduced*	*Documentation last modified*
modhysn	2.1	January 4, 2001

Structure

typedef struct {	
unsigned char *priority;*	Priority of each input buffer When the wave parameter priority is set for w_pri, the priority of the voice being generated will be priority + w_pri Maximum priority: 255
unsigned char *maxPolyphony;*	Max. number of voices used by this input (default 48)

unsigned char *portMode;*	Mode of the stream used by this port
	sceHSynModeHSyn = MIDI stream (default)
	sceHSynModeSESyn = SE stream

unsigned char *waveType;*	Chunk within the bank binary data used by this port
	sceHSynTypeProgram = Use program chunk (default)
	sceHSynTypeTimbre = Use Timbre Chunk (SE)

int *lfoWaveNum;*	Number of user-defined LFOs
sceHSynUserLfoWave **lfoWaveTbl;*	Leading address of user-defined LFO array
int *velocityMapNum;*	Number of user-defined velocity conversion tables
sceHSynUserVelocityMap **velocityMapTbl;*	Leading address of user-defined velocity conversion table array
unsigned char *system*[**sceHSynEnvSize**]*;*	Internal variable area used by this module
} sceHSynEnv;	

Description

Environment buffer which controls the playback state, etc. of every input buffer.

sceHSynUserLfoWave

User-defined LFO waveform

Library	*Introduced*	*Documentation last modified*
modhysn	2.1	January 4, 2001

Structure

typedef struct {	
unsigned char *id;*	ID of LFO specified by Wave Parameter
unsigned short *waveLen;* // in sample	Number of waveform data values Handled in 16-bit units (For a 20-byte waveform: 10)
short **wave;*	Waveform data Signed 16-bit value
} sceHSynUserLfoWave;	

Description

Defines user-defined LFO.

sceHSynUserVelocityMap

User-defined velocity conversion table

Library	*Introduced*	*Documentation last modified*
modhysn	2.1	January 4, 2001

Structure

typedef struct {

    unsigned char *velMap* **[sceHSynNumVelocity];** Velocity == value corresponding to v (1-127)

} sceHSynUserVelocityMap;

Description

Table which modifies Velocity of Note On Message.

Functions

sceHSyn_ MSGetVoiceEnvelopeByID

Find the envelope values of the voices generated by the MIDI stream from the ID

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

Syntax

int sceHSyn_ MSGetVoiceEnvelopeByID (

sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned char *id,*	ID number
unsigned short *ret* **[max_voices],**	Envelope values of voices for the specified ID number
char *max_voices***)**	Maximum number of voices to find

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 returns in *ret* the current envelope values (*) of the voices that were used to generate sound with the specified ID, for the input buffer of the specified MIDI input stream. The maximum value max_voices is the upper limit on the number of voices. The format of ret is the same as the voice_env member of the sceHSyn_VoiceStat structure.

ret is a user-defined array having at least max_voices elements.

(*) The state when sceHSyn_ATick() was called just prior to this function.

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

Return value

>=0	Number of voices that were found
< 0	Error

sceHSyn_ MSGetVoiceStateByID

Find the state of voices generated by the MIDI stream from the ID

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

Syntax

int sceHSyn_ MSGetVoiceStateByID (	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned char *id,*	ID number
unsigned char *ret* **[max_voices],**	Envelope values of voices for the specified ID number
char *max_voices***)**	Maximum number of voices to find

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 returns in *ret* the voice states (*) of the voices that were used to generate sound with the specified ID, for the input buffer of the specified MIDI input stream. The maximum value max_voices is the upper limit on the number of voices. The format of ret is the same as the voice_state member of the sceHSyn_VoiceStat structure.

ret is a user-defined array having at least max_voices elements.

(*) The state when sceHSyn_ATick() was called just prior to this function.

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

Return value

>=0	Number of voices that were found
< 0	Error

sceHSyn_AllNoteOff

Sets all voices of an input buffer to KEY_OFF

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

Syntax

int sceHSyn_AllNoteOff(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number***)**	Input port 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

Sets all voices of the specified input buffer to the KEY_OFF state.

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

Return value

If processing was successful: 0

sceHSyn_AllSoundOff

Mute all voices of an input buffer

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

Syntax

int sceHSyn_AllSoundOff(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number***)**	Input port 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

Mutes all voices of the specified input 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 sceHSyn functions.

Return value

If processing was successful: 0

sceHSyn_ATick

Interrupt processing

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

Syntax

int sceHSyn_ATick(	
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

Called from an interrupt at regular intervals. Processes all input messages and updates the Voice state.

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

Return value

If processing was successful: 0

sceHSyn_GetChStat

Get voice usage state for all channels

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

Syntax

int sceHSyn_GetChStat(

sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
sceHSynChStat **buff_addr***)**	State acquisition buffer 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

Checks the voice usage state for all channels of the specified port.

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

Return value

If processing was successful: 0

sceHSyn_GetReservVoice

Get reserved voice status

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

Syntax

int sceHSyn_GetReservVoice(

unsigned int *voice_bit[2]***)**	This function places the reserved voice of core 0 in *voice_bit[0]* and the reserved voice of core 1 in *voice_bit[1].* Bit 0 corresponds to voice 0, and bit N corresponds to voice N. A voice for which the corresponding bit is set to 1 is considered to be a reserved voice.

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function gets the status of reserved voices which had been previously set with sceHSyn_SetReservVoice. Reserved voices cannot be used by the synthesizer module.

Return value

Always 0

sceHSyn_GetVolume

Get volume value of each input

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

Syntax

unsigned short sceHSyn_GetVolume(	
sceCslCtx **module_context*,	Module Context address
unsigned int *port_number***)**	Input port number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Gets the volume value of an individual input buffer.

Return value

Volume value of specified input buffer

sceHSyn_Init

Initialization

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

Syntax

int sceHSyn_Init(

sceCslCtx **module_context,*	Module Context address
unsigned int *interval***)**	Interval between ATick() calls expressed 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-enabled state)

Description

Initializes the Hardware Synthesizer Module's internal environment and the SPU2.

Effect is set to OFF.

Return value

If processing was successful: 0

sceHSyn_Load

Registers wave data and headers in the SPU2

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

Syntax

int sceHSyn_Load(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
void **spu2_wave_address,*	Wave data address in the SPU2
void **header_address,*	Header address
unsigned int *bank***);**	Bank no. (0-15)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Registers the wave data and headers (parameters) that were transferred to the SPU2.

Proper operation cannot be ensured if the port bank is changed during voice generation.

The wave data that has been transferred to SPU2 and the header (parameter) are registered.

If the port bank is changed during sound generation, then operation is not guaranteed.

Moreover, regarding the attributes of the input environment for the specified input port, when SE stream (sceHSynModeSESyn) is specified for the stream mode (portMode) and when Timbre Chunk (sceHSynTypeTimbre) is specified for Chunk (waveType), the bank number setting is ignored and the wave data that was specified last along with the header, are always used for that input port.

Return value

If processing was successful: 0

sceHSyn_ResetAllControler

Initialize input buffer controller values

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

Syntax

int sceHSyn_ResetAllControler(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number***)**	Input port 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

Restores the values of the specified input buffer's controller to their initial values. The controller values to be restored are:

Hold
Pitch bend
Modulation
Portamento

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

Return value

If processing was successful: 0

sceHSyn_SEAllNoteOff

Set all voices in the SE input buffer to KEY_OFF state

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

Syntax

int sceHSyn_SEAllNoteOff(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number***)**	Input port 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 sets all voices in the input buffer of the specified input SE stream to KEY_OFF state.

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

Return value

If processing was successful: 0

sceHSyn_SEAllSoundOff

Turn off the sound of all voices in the SE input buffer

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

Syntax

int sceHSyn_SEAllSoundOff(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number***)**	Input port 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 turns off or releases the sound of all voices in the input buffer of the specified input SE stream.

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

Return value

If processing was successful: 0

sceHSyn_SERetrieveAllSEMsgIDs

Find all SE message IDs used by the active voices of an SE stream

Library	*Introduced*	*Documentation last modified*
modhysn	2.3.3	October 11, 2001

Syntax

int sceHSyn_SERetrieveAllSEMsgIDs(

sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Input port number
int **ret,*	SE message IDs used by voice
int *max_voices***)**	Maximum number of SE message IDs to retrieve

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

Returns in *ret*, all of the SE message IDs for which sound generation processing is being performed for the input buffer of the specified input stream.

"max_voices" indicates the upper limit on the number of IDs returned.

"ret" is a user array that has at least "max_voices" elements.

Even if the SE stream is being played, depending on the data, there will be states in which even a single voice cannot be allocated. Consequently, there may be situations where all of the SE message IDs that are in use by an active SE sequence cannot be retrieved. Whether SE sequence playback is active should be checked using sceSESq_SeqIsInPlay().

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

Return value

>=0	Number of SE message IDs found
< 0	Error

sceHSyn_SERetrieveVoiceNumberByID

Find the voice numbers of the voices generated by the SE stream from the ID

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

Syntax

int sceHSyn_SERetrieveVoiceNumberByID(

sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned int *id,*	ID number
char **ret,*	Voice numbers of voices for which the specified ID number is being used
char *max_voices)*	Maximum number of voices to find

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 returns in *ret* the voice numbers (0 to 47; 24 and higher are for CORE1) of the voices that were used to generate sound with the specified ID, for the input buffer of the specified SE input stream. The maximum value *max_voices* is the upper limit on the number of voices.

ret is a user-defined array having at least *max_voices* elements.

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

Return value

>=0	Number of voices that were found
< 0	Error

sceHSyn_SESetMaxVoices

Set the upper limit of the total number of voices for which sound is generated by the SE stream

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

Syntax

int sceHSyn_SESetMaxVoices(	
unsigned char *max_voices***)**	Upper limit of the total number of voices for which sound is generated by the SE stream, or 0

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 upper limit of the total number of voices for which sound is generated by the SE stream. Voices are assigned for the SE stream and processing is performed within the range described by this upper limit.

If 0 is specified for *max_voices*, all voices are subject to processing, and voices are assigned according to free voices and priorities.

Operation is not guaranteed if this function is called during voice generation.

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

Return value

If processing was successful: 0

sceHSyn_SetEffectAttr

Set EFFECT

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

Syntax

int sceHSyn_SetEffectAttr(	
sceHSyn_EffectAttr **effect_attribute*)	State of effect to be set

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 SPU2 effect.

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

Return value

If processing was successful: 0

sceHSyn_SetOutputMode

Switch output mode between monaural and stereo

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

Syntax

int sceHSyn_SetOutputMode(	
int *output_mode***)**	Output mode sceHSynOutputMode_Mono Panpots are disabled, and all panpots will be centered sceHSynOutputMode_Stereo Panpots are enabled

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Sets the output mode (panpots enabled or disabled).

Return value

If processing was successful: 0

sceHSyn_SetReservVoice

Set reserved voice

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

Syntax

int sceHSyn_SetReservVoice(

unsigned int *voice_bit[2]***)**	For *voice_bit[0],* specify a core 0 reserved voice, and for *voice_bit[1],* specify a core 1 reserved voice. bit-0 corresponds to voice 0, and bit-N corresponds to voice N. A voice for which the relevant bit is 1 is the reserved voice.

Description

Reserves some of each core's voices and prohibits their use in synthesizer modules. Proper operation cannot be ensured if this function is called while a voice is being generated.

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

Return value

If processing was successful: 0

sceHSyn_SetVoiceStatBuffer

Register Synthesizer status monitor buffer

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

Syntax

int sceHSyn_SetVoiceStatBuffer(	
sceHSyn_VoiceStat **status_buffer*)	Status storage buffer 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 the module's current status monitor buffer.

Status updating depends on ATick() execution.

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

Return value

If processing was successful: 0

sceHSyn_SetVolume

Set volume of each input

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

Syntax

int sceHSyn_SetVolume(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned short *vol***)**	Volume value

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 for an individual input buffer.

If the volume of a voice is set for v_vol, the value that is actually output will be (v_vol * vol) / sceHSyn_Volume_0db.

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

Return value

If processing was successful: 0

sceHSyn_VoiceTrans

Transfer wave data to the SPU2

Library	*Introduced*	*Documentation last modified*
modhysn	2.1	July 2, 2001

Syntax

int sceHSyn_VoiceTrans(

short *channel,*	Channel to be used
unsigned char **data_address,*	Address in data memory (transfer source)
unsigned char **spu2_address,*	SPU2 address (transfer destination)
unsigned int *size*)	Transfer size

Calling conditions

Can be called from a thread

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

Description

Transfers (DMA) wave data to the SPU2.

If data is updated during SPU2 voice generation, the voice which was output using the original data cannot be ensured. (Finer control can be achieved by using libsd.)

Since the current implementation is not multithread safe and the function must be called in an interrupt-enabled state, be sure not to call this function between multiple threads at the same time.

Return value

If processing was successful: 0

Chapter 4: CSL MIDI Stream Generation
Table of Contents

Structures

sceMSInHsMsg

Extended MIDI message

Library	*Introduced*	*Documentation last modified*
modmsin	1.3	February 29, 2000

Structure

typedef struct {
    unsigned char *d[7];*
} sceMSInHsMsg;

Description

This structure is used for extended MIDI messages.

Functions

sceMSIn_Init

Initialization

Library	*Introduced*	*Documentation last modified*
modmsin	1.3	October 11, 2001

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*
modmsin	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.

sceMSIn_MakeHsMsg1

Create extended Pre Voice Control Message (MACRO)

Library	*Introduced*	*Documentation last modified*
modmsin	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.

sceMSIn_MakeHsMsg2

Create extended Voice Control Message (MACRO)

Library	*Introduced*	*Documentation last modified*
modmsin	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.

sceMSIn_MakeHsNoteOff

Create extended Note Off (MACRO)

Library	*Introduced*	*Documentation last modified*
modmsin	1.3	October 11, 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.

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

sceMSIn_MakeHsNoteOn

Create extended Note On (MACRO)

Library	*Introduced*	*Documentation last modified*
modmsin	1.3	October 11, 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.

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

sceMSIn_MakeHsPanpot

Create extended Panpot (MACRO)

Library	*Introduced*	*Documentation last modified*
modmsin	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.

sceMSIn_MakeHsPitchBend

Create extended Pitch Bend (MACRO)

Library	*Introduced*	*Documentation last modified*
modmsin	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.

sceMSIn_MakeHsPreExpression

Create extended Pre Expression (MACRO)

Library	*Introduced*	*Documentation last modified*
modmsin	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.

sceMSIn_MakeHsPrePanpot

Create extended Pre Panpot (MACRO)

Library	*Introduced*	*Documentation last modified*
modmsin	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.

sceMSIn_MakeHsPrePitchBend

Create extended Pre Pitch Bend (MACRO)

Library	*Introduced*	*Documentation last modified*
modmsin	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.

sceMSIn_MakeMsg /3

Pack MIDI message into unsigned int (MACRO)

Library	*Introduced*	*Documentation last modified*
modmsin	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*
modmsin	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 port buffer (MACRO)

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

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.

Return value

When processing is successful: 0

sceMSIn_NoteOn

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

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

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.

Return value

When processing is successful: 0

sceMSIn_NoteOnEx

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

Library	*Introduced*	*Documentation last modified*
modmsin	1.3	October 11, 2001

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 functions or sceHSyn_ATick().

Return value

When processing is successful: 0

sceMSIn_ProgramChange

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

Library	*Introduced*	*Documentation last modified*
modmsin	1.3	October 11, 2001

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 functions or sceHSyn_ATick().

Return value

When processing is successful: 0

sceMSIn_PutExcMsg

Write exclusive message to output port buffer

Library	*Introduced*	*Documentation last modified*
modmsin	1.3	October 11, 2001

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 functions or sceHSyn_ATick().

Return value

When processing is successful: 0

sceMSIn_PutHsMsg

Write extended MIDI message to output port buffer

Library	*Introduced*	*Documentation last modified*
modmsin	1.3	October 11, 2001

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 functions or sceHSyn_ATick().

Return value

When processing is successful: 0

sceMSIn_PutMsg

Write MIDI message to output port buffer

Library	*Introduced*	*Documentation last modified*
modmsin	1.3	October 11, 2001

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 functions or sceHSyn_ATick().

Return value

When processing is successful: 0.

Chapter 5: CSL SE Stream Generation (for IOP)
Table of Contents

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	October 11, 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
unsigned int *prog_number,*	Program 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 message to output port buffer

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

Write pitch LFO message to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.2	October 11, 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
unsigned int *prog_number,*	Program 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 message (specified pitch) to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.2	October 11, 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
unsigned int *prog_number,*	Program 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 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.

Return value

0 if successful

sceSEIn_MakeTimePanpot

Write time pan pot message to output port buffer

Library	*Introduced*	*Documentation last modified*
modsein	2.2	October 11, 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
unsigned int *prog_number,*	Program 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
unsigned int *prog_number,*	Program 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	October 11, 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
unsigned int *prog_number,*	Program 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

Write Note Off message to output port buffer (MACRO)

Library	*Introduced*	*Documentation last modified*
modsein	2.1	October 11, 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
unsigned int *prog_number,*	Program 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 Note Off message to the specified output port buffer.

Return value

If processing was successful	0

sceSEIn_NoteOn

Write Note On message to output port buffer (MACRO)

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

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 message (pitch specification) to output port buffer (MACRO)

Library	*Introduced*	*Documentation last modified*
modsein	2.1	October 11, 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
unsigned int *prog_number,*	Program 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 (pitch specification) 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 6: CSL Software Synthesizer
Table of Contents

Structures

sceSSynEnv

Input environment.

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

Structure

typedef struct {

unsigned int *ee_info_addr;*	Address of management information on the EE
unsigned int *ee_buff_addr;*	Receive buffer address in the EE
unsigned int *ee_buff_length;*	Receive buffer size in the EE
unsigned int *atickCount;*	Transmit data frequency to the EE
unsigned int *ee_buff_write_index;*	Receive buffer write address in the EE
unsigned int *ee_buff_read_index;*	Receive buffer read address in the EE
unsigned char *alignment_adjust_buff[16];*	Alignment adjustment buffer for DMA transfers
sceSifDmaData *dma[4];*	DMA control buffer

} sceSSynEnv;

Description

Environment buffer for managing information such as the state of communication with the EE for each input buffer.

The alignment must equal an integer multiple of 4.

Functions

sceSSyn_ATick

Interrupt processing

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

Syntax

int sceSSyn_ATick

(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

Called from an interrupt at regular intervals.
Transmits data that is in the input buffer to the EE.

Return value

If processing was successful: 0

sceSSyn_Init

Initialization

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

Syntax

int sceSSyn_Init(	
sceCslCtx **module_context,*	Module Context address
unsigned int *interval***)**	Interval between ATick() calls expressed 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

Performs initialization tasks such as reserving the communication line for communicating with the EE.

Return value

If processing was successful: 0

sceSSyn_Load

Read data

Library	*Introduced*	*Documentation last modified*
modssyn	2.2	March 26, 2001

Syntax

int sceSSyn_Load(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number***)**	Input port number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Formal implementation only. No real processing is performed.

Return value

When processing is successful: 0

Chapter 7: CSL MIDI Sequencer
Table of Contents

Structures

sceMidiEnv

Sequence Data environment.

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

Structure

typedef struct {	
unsigned int *songNum;*	SongChunk number that is currently being performed or has been selected
unsigned int *midiNum;*	MidiChunk number that is currently being performed or has been selected
unsigned int *position;*	Current position of Sequence Data (units: ticks)
unsigned int *status;*	Performance status sceMidiStat_ready: Initialized bit sceMidiStat_inPlay: Performance in progress bit sceMidiStat_dataEnd: End of data reached bit sceMidiStat_noLoop: Loop message ignored bit If this bit is set to 1, a loop message within the data is ignored.
unsigned short *outPort[sceMidiNumMidiCh];*	Per-channel output port specification Which channel is output to which port can be specified. Setting is bit mask, so one channel can be output to multiple ports.
unsigned short *excOutPort;*	Exclusive output port. Setting value is bit mask.
unsigned int *(*chMsgCallBack)***(unsigned int, unsigned int);**	Channel message callback
unsigned int *chMsgCallBackPrivateData;*	Channel message callback data
Bool *(*metaMsgCallBack*)**(unsigned char, unsigned char*, unsigned int, unsigned int);**	Meta event callback
unsigned int *metaMsgCallBackPrivateData;*	Meta event callback data
Bool *(*excMsgCallBack*)**(unsigned char*, unsigned int, unsigned int);**	Exclusive callback
unsigned int *excMsgCallBackPrivateData;*	Exclusive callback data
Bool *(*repeatCallBack*)**(sceMidiLoopInfo*, unsigned int);**	Loop control callback
unsigned int *repeatCallBackPrivateData;*	Loop control callback data
unsigned char *system[sceMidiEnvSize];*	Sequencer Module internal variable area
} sceMidiEnv;	

Description

Environment buffer for managing the musical performance state for each Sequence Data buffer.

sceMidiLoopInfo

LOOP (Repeat): Callback information

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

Structure

typedef struct {	
unsigned char *type;*	LOOP(Repeat) generated chunk type sceMidiLoopInfoType_Midi: Midi Chunk sceMidiLoopInfoType_Song: Song Chunk
unsigned char *loopTimes;*	Loop frequency within loop message (0 indicates unlimited looping)
unsigned char *loopCount;*	Loop frequency (when loopTimes == 0: undefined)
unsigned int *loopId;*	Loop identifier
} sceMidiLoopInfo;	

Description

Structure used in loop control callback arguments.

Functions

chMsgCallBack

Channel message callback specification

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 6, 2000

Syntax

unsigned int chMsgCallBack(

unsigned int *message,*	Sequence Command Message
	bit 0-7: status
	bit 8-14: 1st data
	bit 16-22: 2nd data
unsigned int *private_data***)**	chMsgCallBackPrivateData of sceMidiEnv

Description

Specification of the callback function which is set in the environment buffer and is called immediately before sending a channel message.

The message that is actually sent will be the return value of this function. However, no message is sent when the return value is sceMidi_ChMsgNoData.

Return value

Transmit Sequence Command Message

excMsgCallBack

Exclusive message callback specification

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 6, 2000

Syntax

Bool excMsgCallBack(	
unsigned char **exclusive_data,*	Exclusive data address
unsigned int *data_length,*	Exclusive data byte count
unsigned int *private_data***)**	excMsgCallBackPrivateData of sceMidiEnv

Description

Specification of the callback function which is set in the environment buffer and controls the transmission of exclusive messages.

Return value

True: The exclusive message was transmitted.

False: The exclusive message was not transmitted.

metaMsgCallBack

Meta event callback specification

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 6, 2000

Syntax

Bool metaMsgCallBack(

unsigned char *meta_number,*	Meta event number
unsigned char **meta_data,*	Meta event data address
unsigned int *data_length,*	Meta event data byte count
unsigned int *private_data)*	metaMsgCallBackPrivateData of sceMidiEnv

Description

Specification of the callback function which is set in the environment buffer and controls meta event processing.

Return value

True: This meta event was processed by the Sequencer Module.

False: This meta event was not processed by the Sequencer Module.

repeatCallBack

Loop control callback specification

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 6, 2000

Syntax

Bool repeatCallBack(	
sceMidiLoopInfo **loop_information,*	Loop information
unsigned int *private_data***)**	repeatCallBackPrivateData of sceMidiEnv

Description

Specification of the callback function which is set in the environment buffer and controls loops.

Return value

True: Looping (repeating) was performed.

False: Looping (repeating) was not performed.

sceMidi_ATick

Interrupt processing

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 11, 2001

Syntax

int sceMidi_ATick(	
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

Called from an interrupt at regular intervals. It advances the performance by a tickInterval when the environment for which the performance is in progress is sceMidiEnv.

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

Return value

If processing was successful: 0

sceMidi_GetEnv

Get the environment address (macro)

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

Syntax

sceMidiEnv *sceMidi_GetEnv(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number***)**	Input port number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Gets the environment address which corresponds to port_number.

Return value

Environment address

sceMidi_GetTempo

Get performance tempo from relative and absolute tempos (MIDI)

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

Syntax

unsigned int sceMidi_GetTempo(	
unsigned char *a_tempo,*	Absolute tempo
unsigned short *r_tempo)*	Relative tempo

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Gets the performance tempo from the absolute and relative tempos.

Return value

Performance tempo.

sceMidi_Init

Initialization

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

Syntax

int sceMidi_Init(	
sceCslCtx **module_context,*	Module Context address
unsigned int *interval***)**	Interval between ATick() calls expressed 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 the MIDI Sequencer Module's internal environment.

Return value

If processing was successful: 0

sceMidi_isDataEnd

Get environment status (at end of data or not?) macro

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

Syntax

unsigned int sceMidi_isDataEnd(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number***)**	Input port number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Inquires whether the end of data was reached for the specified environment.

Return value

Non-zero: End of data was reached

sceMidi_isInPlay

Get environment status (is performance in progress or not?) macro

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

Syntax

unsigned int sceMidi_isInPlay(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number***)**	Input port number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Inquires whether the performance is in progress for the specified environment.

Return value

Non-zero: Performance is in progress

sceMidi_Load

Read sequence data

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

Syntax

int sceMidi_Load(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number***)**	Input port number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Reports sequence data updates.

When the performance is in progress for the specified environment, data updates to that environment and calls to sceMidi_Load() are not permitted.

Return value

If processing was successful: 0

sceMidi_MidiGetAbsoluteTempo

Get absolute tempo (MIDI)

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

Syntax

unsigned char sceMidi_MidiGetAbsoluteTempo(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number***)**	Input port number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Gets the absolute tempo.

Return value

Absolute tempo

sceMidi_MidiGetRelativeTempo

Get relative tempo (MIDI)

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

Syntax

unsigned short sceMidi_MidiGetRelativeTempo(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number***)**	Input port number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Gets the relative tempo.

Return value

Relative tempo

sceMidi_MidiGetUSecTempo

Get tempo in microseconds (MIDI)

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

Syntax

unsigned char sceMidi_MidiGetUSecTempo(	
sceCslCtx **module_context*,	Address of Module Context
unsigned int *port_number***)**	Input port number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function gets the current tempo in microseconds. The return value represents the length of a quarter note in microseconds.

Return value

Tempo in microseconds

sceMidi_MidiPlaySwitch

Start/stop performance (MIDI)

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 11, 2001

Syntax

int sceMidi_MidiPlaySwitch(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
int *command***)**	sceMidi_MidiPlayStop: stops performance
	sceMidi_MidiPlayStart: starts performance

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

Starts or stops the performance.

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

Return value

If processing was successful: 0

sceMidi_MidiSetAbsoluteTempo

Change absolute tempo (MIDI)

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 11, 2001

Syntax

int sceMidi_MidiSetAbsoluteTempo(

sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned char *tempo***)**	Tempo (20 - 255)

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

Changes the tempo.

Equivalent to a tempo meta event.

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

Return value

If processing was successful: 0

sceMidi_MidiSetLocation

Change performance position (MIDI)

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 11, 2001

Syntax

int sceMidi_MidiSetLocation(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned int *position***)**	Position within sequence data (Tick)

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

Changes the position within the sequence data.

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

Return value

If processing was successful: 0

sceMidi_MidiSetRelativeTempo

Change relative tempo (MIDI)

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 11, 2001

Syntax

int sceMidi_MidiSetRelativeTempo(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned short *tempo***)**	Relative tempo (if set to sceMidi_RelativeTempoNoEffect: No effect)

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 relative tempo.

If the absolute tempo is a_tempo and the relative tempo is r_tempo, then the performance tempo, which is represented by *tempo*, will be:

tempo = (a_tempo * r_tempo) / sceMidi_RelativeTempoNoEffect

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

Return value

If processing was successful: 0

sceMidi_MidiSetUSecTempo

Set tempo in microseconds (MIDI)

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

Syntax

int sceMidi_MidiSetUSecTempo(	
sceCslCtx **module_context*,	Address of Module Context
unsigned int *port_number*,	Input port number
unsigned short *tempo*)	Tempo (in microseconds)

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 sets the tempo in microseconds.

tempo should be specified as a value that represents the length of a quarter note in microseconds.

Although fine tempo control can be achieved using this function, since the parsing of score is ultimately quantized at the resolution with which sceMidi_ATick is called, be sure to take this into consideration when setting the tempo.

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

Return value

When processing is successful, zero is returned.

sceMidi_MidiSetVolume

Change (absolute) channel volume (MIDI)

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 11, 2001

Syntax

int sceMidi_MidiSetVolume(

sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input number
unsigned char *ch,*	Channel (0-15) sceMidi_MidiSetVolumeMasterVol: Master volume
unsigned char *vol***)**	Volume (sceMidi_Volume0db: No change)

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 relative volume of the channel.

If the channel volume is ch_vol, the master volume is m_vol, and the relative volume is r_vol, then the volume that is output, which is represented by vol, will be:

vol = (ch_vol * m_vol * r_vol) / (sceMidi_Volume0db*sceMidi_Volume0db)

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

Return value

If processing was successful: 0

sceMidi_MidiVolumeChange

Change channel volume (MIDI)

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 11, 2001

Syntax

int sceMidi_MidiVolumeChange(

sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned char *ch,*	Channel (0--15)
	255: Treated as if all channels were specified.
unsigned char *vol***)**	Volume (0--127)

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 channel volume.

Equivalent to the volume of a sequence command.

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

Return value

If processing was successful: 0

sceMidi_SelectMidi

Select Midi Block to be performed (MIDI)

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 11, 2001

Syntax

int sceMidi_SelectMidi(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Output port number
unsigned int *midi_block_number***)**	Midi Block 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

Selects the Midi Block to be performed.

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

Return value

If processing was successful: 0

sceMidi_SelectSong

Select Song Block to be performed (SONG)

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 11, 2001

Syntax

int sceMidi_SelectSong(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned int *song_block_number*)	Song Block 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

Selects the Song Block to be performed.

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

Return value

If processing was successful: 0

sceMidi_SongPlaySwitch

Start/stop performance (SONG)

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 11, 2001

Syntax

int sceMidi_SongPlaySwitch(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
int *command***)**	sceMidi_SongPlayStop: Stop the performance of the song
	sceMidi_SongPlayPause: Pause the performance of the song
	sceMidi_SongPlayStart: Start the performance of a song
	sceMidi_SongPlayContinue: Start the performance of a song

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

Starts, stops, or pauses the performance.

When command is set to sceMidi_SongPlayStart or sceMidi_SongPlayContinue, playback of the song will start at the beginning of the Song if it is stopped, or immediately after a Select. If it is paused, playback of the song will start from the paused location.

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

Return value

If processing was successful: 0

sceMidi_SongSetAbsoluteTempo

Change absolute tempo (SONG)

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 11, 2001

Syntax

int sceMidi_SongSetAbsoluteTempo(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned char *tempo***)**	Tempo (20 -- 255)

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

Changes the tempo.

Equivalent to a song tempo message.

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

Return value

If processing was successful: 0

sceMidi_SongSetLocation

Set/change song location (SONG)

Library	*Introduced*	*Documentation last modified*
modmidi	1.2	October 11, 2001

Syntax

int sceMidi_SongSetLocation(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Output port number
unsigned int *position,*	Song position
unsigned int *mode)*	Operation mode sceMidi_SSL_Now: Immediately interrupt the song that is being performed, change the position, and restart the song. sceMidi_SSL_Delay: Wait for the end of the current song, change the position, and restart the song. sceMidi_SSL_WithPreCommand: Start (restart) the song beginning with the MIDI song starting command located one command before position. sceMidi_SSL_WithoutPreCommand: Start (restart) the song beginning with the MIDI song starting command indicated by position.

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 or changes the position within a Song.

For position, specify which MIDI song starting command counting from the beginning of the Song Block, that the destination position is to correspond to, where the first MIDI song starting command is counted as 0.

For mode, specify a value obtained by taking the appropriate sum of the four mode constants. However, sceMidi_SSL_Now and sceMidi_SSL_Delay cannot be specified at the same time. Likewise, sceMidi_SSL_WithPreCommand and sceMidi_SSL_WithoutPreCommand cannot be specified at the same time.

sceMidi_SSL_Now and sceMidi_SSL_Delay specify the action to take related to the Song Block that is currently being performed before moving the position. If the song is paused, the position is moved immediately in a similar manner as for sceMidi_SSL_Now for both options, but the song is not restarted.

sceMidi_SSL_WithPreCommand and sceMidi_SSL_WithoutPreCommand are specifications relating to the MIDI command to be executed, after the position is moved. When sceMidi_SSL_WithPreCommand is specified, the MIDI commands are executed up to the MIDI song starting command preceding the MIDI song starting command indicated by position. However, any repeat command within this range is ignored.

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

Return value

If processing was successful: 0

sceMidi_SongSetRelativeTempo

Change relative tempo (SONG)

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 11, 2001

Syntax

int sceMidi_SongSetRelativeTempo(

sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned short *tempo***)**	Relative tempo (if set to sceMidi_RelativeTempoNoEffect: No effect)

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 relative tempo. If the absolute tempo is a_tempo and the relative tempo is r_tempo, then the performance tempo, which is represented by *tempo*, will be:

tempo = (a_tempo * r_tempo) / sceMidi_RelativeTempoNoEffect

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

Return value

If processing was successful: 0

sceMidi_SongSetVolume

Change (relative) volume (SONG)

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 11, 2001

Syntax

int sceMidi_SongSetVolume(

sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned char *vol***)**	Volume (if set to sceMidi_Volume0db: No change)

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 relative volume of the song.

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

Return value

If processing was successful: 0

sceMidi_SongVolumeChange

Change volume (SONG)

Library	*Introduced*	*Documentation last modified*
modmidi	1.1	October 11, 2001

Syntax

int sceMidi_SongVolumeChange(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned char *vol***)**	Volume (0-128)

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.

Equivalent to the volume of a song command.

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

Return value

If processing was successful: 0

Chapter 8: CSL MIDI Monophonic
Table of Contents

Structures

sceMidiMono_Env

Environment

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

Structure

#define sceMidiMono_MaxKey: 128

#define sceMidiMono_MaxCh: 16

typedef struct {

unsigned char *mono***[sceMidiMono_MaxCh];**	sceMidiMonoOn: Monophonic is assigned. sceMidiMonoOff: Monophonic is not assigned.
unsigned char *onKey***[sceMidiMono_MaxCh];**	Mono Module internal variable
unsigned char *velocity***[sceMidiMono_MaxCh];**	Mono Module internal variable
unsigned char *key* **[sceMidiMono_MaxCh][sceMidiMono_MaxKey];**	Mono Module internal variable

} sceMidiMono_Env;

Description

Environment buffer which specifies the processing state to every input buffer and performs management.

Functions

sceMidiMono_ATick
Interrupt processing

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

Syntax

int sceMidiMono_ATick(	
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

Called from an interrupt at regular intervals.

Return value

When processing is successful: 0

sceMidiMono_GetEnv

Get environment address

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

Syntax

sceMidiMono_Env *sceMidiMono_GetEnv(

sceCslCtx **module_context,*	Module Context address
unsigned int *port_number***)**	Input port number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Gets the environment address corresponding to the port_number.

Return value

Environment address

sceMidiMono_Init

Initialization

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

Syntax

int sceMidiMono_Init(	
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

Initializes the internal environment of the MIDI monophonic module.

Return value

If processing was successful: 0

sceMidiMono_SetMono

Monophonic assignment switch

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

Syntax

int sceMidiMono_SetMono(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
unsigned char *channel,*	MIDI channel (0-15)
int *switch***)**	sceMidiMonoOn: Assign as monophonic.
	sceMidiMonoOff: Do not assign as monophonic.

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

Specifies how each channel will be assigned.

Return value

If processing was successful: 0

Chapter 9: Low-Level Sound Library
Table of Contents

Structures

sceSdBatch

Batch command

Library	*Introduced*	*Documentation last modified*
libsd	1.1	October 6, 2000

Structure

typedef struct {

u_short *func;*	Set any one of the following functions:
	SD_BSET_PARAM 0x01 Executes sceSdSetParam.
	SD_BGET_PARAM 0x10 Executes sceSdGetParam.
	SD_BSET_SWITCH 0x02 Executes sceSdSetSwitch.
	SD_BGET_SWITCH 0x12 Executes sceSdGetSwitch.
	SD_BSET_ADDR 0x03 Executes sceSdSetAddr.
	SD_BGET_ADDR 0x13 Executes sceSdGetAddr.
	SD_BSET_CORE 0x04 Executes sceSdSetCoreAttr.
	SD_BGET_CORE 0x14 Executes sceSdGetCoreAttr.
	SD_WRITE_IOP 0x05 Writes to IOP memory.
	SD_WRITE_EE 0x06 Writes to EE memory.
	SD_RETURN_EE 0x07 Transfers "returns" to EE memory.
u_short *entry;*	Entry passed to *func*. For the wrapper API, it corresponds to the first argument.
u_int *value;*	Value passed to *func*. For the wrapper API, it corresponds to the second argument.

} *sceSdBatch;*

Description

This structure displays batch commands. The structure's array is passed to the batch processing API as a batch command string.

When SD_WRITE_IOP is specified as func, the value of entry is written to the IOP memory address specified in value.

When SD_WRITE_EE is specified as func, the value of entry is written to the EE memory address specified in value. SIF DMA is used internally.

When SD_RETURN_EE is specified as func, the "returns" (see sceSdProcBatch() for the returned value array) is transferred to the EE memory address specified in value and in only the number of bytes indicated in entry. SIF DMA is used internally.

If SD_BSET_* is specified in func, make sure that the register to be ultimately processed is not specified more than once. Only one SD_BSET_CORE should be included in a single call.

See also

sceSdProcBatch(), sceSdProcBatchEx()

sceSdEffectAttr

Effect attributes

Library	*Introduced*	*Documentation last modified*
libsd	1.1	October 11, 2001

Structure

typedef struct {	
int *core;*	Core specification (currently unused)
int *mode;*	Effect mode
short *depth_L;*	Effect return volume (depth/left)
short *depth_R;*	Effect return volume (depth/right)
int *delay;*	Delay time (ECHO, DELAY only)
int *feedback;*	Feedback (ECHO only)
} sceSdEffectAttr;	

Description

This structure is used for setting the effect attributes.

<mode>

mode specifies the mode of the effect. The valid modes and the amount of space required in sound memory are as follows:

Table 9-1

Macro	Type	Size (bytes)
SD_REV_MODE_OFF	Off	0x80
SD_REV_MODE_ROOM	Room	0x26c0
SD_REV_MODE_STUDIO_A	Studio (small)	0x1f40
SD_REV_MODE_STUDIO_B	Studio (medium)	0x4840
SD_REV_MODE_STUDIO_C	Studio (large)	0x6fe0
SD_REV_MODE_HALL	Hall	0xade0
SD_REV_MODE_SPACE	Space echo	0xf6c0
SD_REV_MODE_ECHO	Echo	0x18040
SD_REV_MODE_DELAY	Delay	0x18040
SD_REV_MODE_PIPE	Pipe echo	0x3c00

When SD_REV_MODE_CLEAR_WA is ORed together with another mode setting, the effect area is cleared when the mode is set.

Use DMA channel 0 for clearing. To specify the DMA channel during a clear, use sceSdClearEffectWorkArea().

If the transfer interrupt handler is specified in DMA channel 0, the handler is saved during clear processing and it cannot be called even after clearing has completed. At that time, the system waits internally for the DMA transfer to end, so it is not necessary to check the status with sceSdVoiceTransStatus().

<depth>

The effect return volume (depth) is set independently for the left and right, within the range -0x8000 to 0x7fff. If the specified value is negative, the phase of the effect component (i.e., wet) will be inverted. The specified value is used as the setting for the basic parameter registers SD_P_EVOLL/SD_P_EVOLR.

<delay>

Valid only for ECHO and DELAY. The delay time should be specified within the range 0-127.

<feedback>

Valid only for ECHO and DELAY. The feedback value should be specified within the range 0-127.

See also

sceSdSetEffectAttr(), sceSdGetEffectAttr()

Functions

sceSdBlockTrans

Transfer to I/O block

Library	*Introduced*	*Documentation last modified*
libsd	1.1	October 11, 2001

Syntax

int sceSdBlockTrans (

short *channel,*	Transfer channel. 0 or 1 can be specified.
u_short *mode,*	Transfer mode
u_char **m_addr,*	IOP memory-side address
u_int *size***[**	Transfer size
u_char **start_addr***])**	Absolute address where transfer starts in IOP memory (only when SD_TRANS_MODE_WRITE_FROM is specified for mode. Can be omitted if nothing between the brackets [ ] is specified.)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

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

Description

Performs transfers related to input/output blocks of the SPU2.

Bit mask that can be set for mode

Transfer direction

SD_TRANS_MODE_WRITE	0
SD_TRANS_MODE_READ	1
SD_TRANS_MODE_STOP	2
SD_TRANS_MODE_WRITE_FROM	3

Transfer setting (WRITE/READ only)

SD_BLOCK_ONESHOT (0<<4)
SD_BLOCK_LOOP (1<<4)

Transfer starting block (READ only)

SD_BLOCK_C0_VOICE1
SD_BLOCK_C0_VOICE3
SD_BLOCK_C1_SINL
SD_BLOCK_C1_SINR
SD_BLOCK_C1_VOICE1

SD_BLOCK_C1_VOICE3
SD_BLOCK_C0_MEMOUTL
SD_BLOCK_C0_MEMOUTR
SD_BLOCK_C0_MEMOUTEL
SD_BLOCK_C0_MEMOUTER
SD_BLOCK_C1_MEMOUTL
SD_BLOCK_C1_MEMOUTR
SD_BLOCK_C1_MEMOUTEL
SD_BLOCK_C1_MEMOUTER

Number of transfer blocks (READ only)

SD_BLOCK_COUNT(x) ((x)<<12)

The data format employed at the IOP side is 16-bit, little endian, signed straight PCM. Moreover, with the current specifications, the left and right channels must be interleaved every 512 bytes.

If SD_TRANS_MODE_WRITE is specified for mode, data is transferred from IOP memory to the input block. If SD_TRANS_MODE_READ is specified for mode, data is transferred to IOP memory from the output block that was specified by mode.

If SD_TRANS_MODE_WRITE_FROM is specified for mode, the transfer starts from the location in IOP memory that was specified by start_addr. The location specified by start_addr must be in the (m_addr + size) area within IOP memory. Otherwise, this is the same as SD_TRANS_MODE_WRITE.

start_addr is referenced only when SD_TRANS_MODE_WRITE_FROM is specified for mode. If another transfer direction is specified, start_addr need not be specified (that is, there will only be four arguments).

If SD_TRANS_MODE_STOP is specified for mode, the transfer is interrupted. At this time, the return value is equivalent to that of sceSdBlockTransStatus(). For specifications about this return value, see the description of sceSdBlockTransStatus().

If SD_BLOCK_ONESHOT is specified for mode, the waveform data for the range that was set will be performed only once. When the performance ends, waveform data that remains in the buffer within the SPU2 is played in a loop. To stop this, use an interrupt or polling to detect the end of the playback and execute SD_TRANS_MODE_STOP.

If SD_BLOCK_LOOP is specified, the waveform data for the range that was set will be performed repeatedly. In this case, size must be a multiple of 1024.

Also, if SD_BLOCK_ONESHOT is specified for mode, an interrupt will occur when the endpoint of the IOP-side buffer is accessed. If SD_BLOCK_LOOP is specified for mode, an interrupt will occur when an intermediate point and the endpoint of the IOP-side buffer are accessed.

For the number of transfer blocks, specify a numerical value shifted left by 12 bits. Since the transfer blocks are arranged in order at the "transfer starting block," if you want to transfer SD_BLOCK_C0_MEMOUTL and SD_BLOCK_C0_MEMOUTR, specify SD_BLOCK_C0_MEMOUTL for the transfer starting block and specify (2<<12) for the number of transfer blocks. The size of one block is 1 kilobyte. These settings are unnecessary for SD_TRANS_WRITE(_FROM).

Although each block is buffered in a 512-byte double buffer, both double buffers are transferred to IOP memory during a READ. As a result, one of the buffers is disabled because it has old data or is being rewritten. Furthermore, during a READ, buffer switching is captured by an SPU2 interrupt, and this function will be called from the SPU2 interrupt handler. The transfer will begin when the buffer is switched after this function call, so the address that caused the SPU2 interrupt should be used to determine which buffer is valid.

Notes

When the data transfer direction is SPU2 local memory => IOP memory during a USB isochronous transfer, the operation on the USB side will timeout and cannot be performed properly.

Return value

Number of bytes that were transferred. Negative value if an error occurred.

If SD_TRANS_MODE_STOP was specified for mode, this will be the position that was being accessed at that time plus buffer information.

sceSdBlockTransStatus

Get status of I/O block transfer

Library	*Introduced*	*Documentation last modified*
libsd	1.1	July 2, 2001

Syntax

u_int sceSdBlockTransStatus (	
short *channel,*	Transfer channel. 0 or 1 can be specified.
short *flag***)**	Status flag (unimplemented)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Gets the status of an I/O block transfer. Bits 0~23 of the return value represent the address (in IOP memory) during an access.

The value becomes 0 when the transfer ends.

Bit 24, the buffer number during a transfer, has significance only in the case of SD_BLOCK_LOOP. During the transfer of the first and second halves of the buffer, 0 and 1, respectively, are returned.

Bits 25-31 are a reserved area, which may be used in the future.

Return value

Transfer status

sceSdClearEffectWorkArea

Clear the effect work area

Library	*Introduced*	*Documentation last modified*
libsd	1.1	October 11, 2001

Syntax (IOP)

int sceSdClearEffectWorkArea(

int *core,*	Specifies the core. (0 or 1)
int *channel,*	Specifies the DMA channel used for clearing. (0 or 1)
int *effect_mode***)**	Specifies the effect mode.

Calling conditions

Can be called from a thread

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

Description

A clear operation is performed using DMA. Channel specifies the DMA channel to be used.

The core used by the effect area that will be cleared is specified by core.

If the transfer interrupt handler is specified in the specified channel, the handler is saved during clear processing and it cannot be called even after clearing has completed. At that time, the system waits internally for the DMA transfer to end, so it is not necessary to check the status with sceSdVoiceTransStatus().

Return value

None

sceSdGetAddr

Get register wrapper address value

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

Syntax

u_int sceSdGetAddr (

u_short *register)*	Number of register for which the parameter value will be obtained

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Gets the address information held by the specified register.

Use this API for the SD_A_* and SD_VA_* series registers.

Although internal addresses in the SPU2 hardware are represented as short words, specify bytes for this API.

<Syntax for specifying the register number>

```
For SD_A_* :  SD_CORE_? | SD_A_*
For SD_VA_*:  SD_CORE_? | SD_VOICE_?? | SD_VA_*
```

Return value

Value obtained from register (in bytes)

sceSdGetCoreAttr

Get pseudo register wrapper core settings

Library	*Introduced*	*Documentation last modified*
libsd	1.1	July 2, 2001

Syntax

u_short sceSdGetCoreAttr (	
u_short *entry)*	Entry for which the value is to be obtained

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Gets the core setting parameter held by the specific entry.

Although *entry* is not a register, it is used in the same manner as the wrapper API. This function can also be used as a batch command. Use this API for SD_C_* series entries (see below).

Table 9-2

Entry	Contents
SD_C_EFFECT_ENABLE	Enable writing to effect area (0 or 1)
SD_C_IRQ_ENABLE	Enable SPU2 interrupt (0 or 1)
SD_C_MUTE_ENABLE	Mute (0 or 1)
SD_C_NOISE_CLK	Noise generator M-series shift frequency (6 bits)
SD_C_SPDIF_MODE	SPDIF setting (mask)

For the SD_C_*_ENABLE series entries, 1 is returned with Enable and 0 is returned with Disable.

For SD_C_NOISE_CLK, 0 to 63 values are returned.

For SD_C_SPDIF_MODE the logical OR values of the following flags are returned

A core cannot be specified for SD_C_SPDIF_MODE and SPU2 settings will be returned whichever flag is obtained.

Table 9-3

Flag	Meaning
SD_SPDIF_OUT_OFF	Turn off output to SPDIF.
SD_SPDIF_OUT_PCM	Output will be the same as analog output,using PCM (default).
SD_SPDIF_OUT_BITSTREAM	Output the data that was input for the Core0 input block as a bit stream.
SD_SPDIF_OUT_BYPASS	Output the data that was input for the Core0 input block bypassing the internal SPU.
SD_SPDIF_COPY_NORMAL	Normal copy protection (first-generation recordable; Default).
SD_SPDIF_COPY_PROHIBIT	Digital recording prohibited.

Syntax for specifying entry:

For SD_C_* : SD_CORE_? | SD_C_*

Return value

Value obtained from entry

sceSdGetEffectAttr

Get the effect attribute

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

Syntax (IOP)

void sceSdGetEffectAttr (

int *core,*	Specifies the core. (0 or 1)
sceSdEffectAttr **attr***);**	Pointer to effect attribute structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Reads the attributes of the effect.

See the description of sceSdEffectAttr for more information.

Return value

None

sceSdGetParam

Get register wrapper basic parameter

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

Syntax

u_short sceSdGetParam (

u_short *register)*	Number of register for which the parameter value will be obtained

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Gets the 16-bit parameter from the basic parameter registers and the volume registers.

Use this API for SD_P_* and SD_VP_* series registers.

<Syntax for specifying the register number>

For SD_P_* : SD_CORE_? | SD_P_*
For SD_VP_*: SD_CORE_? | SD_VOICE_?? | SD_VP_*

Return value

Value obtained from register.

sceSdGetSpu2IntrHandlerArgument

Get SPU2 interrupt handler data

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

Syntax (IOP)

void* sceSdGetSpu2IntrHandlerArgument (void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function gets a pointer to the data that was registered when the SPU2 interrupt handler was set.

Return value

Pointer to data that was registered when SPU2 interrupt handler was set

See also

sceSdSetSpu2IntrHandler()

sceSdGetSwitch

Get register wrapper voice control parameter

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

Syntax

u_int sceSdGetSwitch (	
u_short *register)*	Number of register for which the parameter value will be obtained

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Gets the on/off flag for each voice from the voice control parameter register.

Use this API for the SD_S_* series registers.

Syntax for specifying the register number:

For SD_S_* : SD_CORE_? | SD_S_*

Return value

Value (bit mask) obtained from the register

sceSdGetTransIntrHandlerArgument

Get transfer interrupt handler data

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

Syntax (IOP)

void* sceSdGetTransIntrHandlerArgument (	
int *channel***);**	Transfer channel. 0 or 1 can be specified.

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function gets a pointer to the data that was registered when the transfer interrupt handler was set.

Return value

Pointer to data that was registered when transfer interrupt handler was set

See also

sceSdSetTransIntrHandler()

sceSdInit

Initialize sound device

Library	*Introduced*	*Documentation last modified*
libsd	1.1	October 11, 2001

Syntax

int sceSdInit (

int *flag***)**	Initialization flag.	
	SD_INIT_COLD	Initialize all
	SD_INIT_HOT	Do not initialize voice, volume and effect settings

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

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

Description

Initializes the sound device. Also makes settings for SPU2-related interrupt controllers at the same time.

Return value

0: normal termination; -1: error

sceSdNote2Pitch

Convert from note value to pitch value

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

Syntax

u_short sceSdNote2Pitch (

u_short *center_note,*	Base note during sampling
u_short *center_fine,*	Fine for base note during sampling
u_short *note,*	Note
short *fine*)	Fine for note

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Calculates the pitch (i.e., the value set in the SPU2 register) from the center note and the generated note.

Since the return value may exceed 0x3fff, you must confirm that the upper limit has not been exceeded and specify the return value for the second argument of sceSdSetParam(SD_VP_PITCH,).

Return value

Pitch

sceSdPitch2Note

Convert from pitch value to note value

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

Syntax

u_short sceSdPitch2Note (	
u_short *center_note,*	Base note during sampling
u_short *center_fine,*	Fine for the base note during sampling
u_short *pitch***)**	Pitch

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

The generated note is calculated from the center note and the generated pitch (i.e., the value set in the SPU2 register).

Return value

Note value (upper 8 bits: note; lower 8 bits: fine)

sceSdProcBatch

Process a batch

Library	*Introduced*	*Documentation last modified*
libsd	1.1	October 11, 2001

Syntax

int sceSdProcBatch (

sceSdBatch* *batch,*	Pointer to batch command structure array
u_int *returns*[],	Address where command's return values are output If null, they are not output.
u_int *num*)	Number of commands in the batch

Calling conditions

Can be called from a thread

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

Description

Batch processes register setting, getting, etc.

See the description of sceSdBatch for more information on batch command types, restrictions, etc.

Note that if this function is called simultaneously in a multithreaded environment so that the same register is set from more than one thread, the actual value that is set will be unpredictable.

Return value

Number of processed commands.

If an error occurred, the ordinal number of the last command processed is converted to a negative number and returned.

sceSdProcBatchEx

Process batch with voice batch processing

Library	*Introduced*	*Documentation last modified*
libsd	1.1	October 11, 2001

Syntax

int sceSdProcBatchEx (

sceSdBatch* *batch,*	Pointer to batch command structure array
u_int *returns***[],**	Address where the command's return value is output. If null, it is not output.
u_int *num*	Number of commands in the batch
u_int *voice***)**	The voice for which voice batch processing is performed, specified using a bit mask.

Calling conditions

Can be called from a thread

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

Description

For commands that specify a voice in the register (e.g. SD_V* series), the command for each voice must be specified in sceSdProcBatch.

However, using sceSdProcBatchEx, the processing of multiple voices can be batched with one command, by specifying the voices in the voice argument voice with a bit mask. In order to enable batch processing, in entry in the batch command data structure it is necessary to specify SD_VOICE_XX by ORing.

(Example: SD_CORE_0|SD_VP_ENVX|SD_VOICE_XX)

The argument num is the number of entries. A command that performs voice batch processing is also counted as 1. On the other hand, the number of returned values is the number of commands actually executed.

After voice batch processing is performed, the commands for each voice are counted individually.

The returns[ ] area contains the values returned after command execution, so the following area is required:

No. of command executions (same as return value num) * 4 bytes

See the description of sceSdBatch for more information on batch command types, restrictions, etc.

Note that if this function is called simultaneously in a multithreaded environment so that the same register is set from more than one thread, the actual value that is set will be unpredictable.

Return value

Number of processed commands.

If an error occurred, the ordinal number of the last command processed is converted to a negative number and returned.

sceSdSetAddr

Set register wrapper address value

Library	*Introduced*	*Documentation last modified*
libsd	1.1	October 11, 2001

Syntax

void sceSdSetAddr(

u_short *register,*	Number of register in which the parameter will be set
u_int *value)*	Parameter value to be set in the register (bytes)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

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

Description

Sets the address in the address-specification register.

Use this API for the SD_A_* and SD_VA_* series registers.

Because of hardware restrictions, the address must be a multiple of 16. If it is not a multiple of 16, the extra bits are ignored.

Note that if this function is called simultaneously in a multithreaded environment so that the same register is set from more than one thread, the actual value that is set will be unpredictable.

Although internal addresses in the SPU2 hardware are represented as short words, specify bytes for this API.

Syntax for specifying the register number:

For SD_A_* : SD_CORE_? | SD_A_*

For SD_VA_*: SD_CORE_? | SD_VOICE_?? | SD_VA_*

SD_VA_NAX is read only, so it cannot be set.

Return value

None

sceSdSetCoreAttr

Set pseudo register wrapper core settings

Library	*Introduced*	*Documentation last modified*
libsd	1.1	October 11, 2001

Syntax

void sceSdSetCoreAttr(

u_short *entry,*	Entry for which the parameter will be set (see below)
u_short *value)*	Parameter value to be set in *entry*

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

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

Description

Sets the core setting parameter value in entry.

Although entry is not a register, it is used in the same manner as the wrapper API. This function can also be used as a batch command. Use this API for SD_C_*_ENABLE series entries.

Table 9-4

Entry	Function
SD_C_EFFECT_ENABLE	Enables writing to effect area (0 or 1).
SD_C_IRQ_ENABLE	Enables SPU2 interrupt (0 or 1).
SD_C_MUTE_ENABLE	Mute (0 or 1)
SD_C_NOISE_CLK	M system shift frequency of noise generator (6 bits)
SD_C_SPDIF_MODE	SPDIF setting (mask)

For entries of the SD_C_*_ENABLE series, set value to 1 to enable, or 0 to disable.

For SD_C_NOISE_CLK, set value in the range 0 to 63.

For SD_C_SPDIF_MODE, set value to the logical OR of the following flags.

Initially, during an SPU2 interrupt, SD_C_IRQ_ENABLE should always have a value of 0 within the interrupt handler. Its value should be set to 1 if another interrupt is needed, when the interrupt handler completes.

Table 9-5

Flag	Meaning
SD_SPDIF_MEDIA_DVD	Media = DVD.
SD_SPDIF_MEDIA_CD	Media = CD. (default)
SD_SPDIF_OUT_OFF	Turn off output to SPDIF.
SD_SPDIF_OUT_PCM	Output will be the same as analog output, using PCM.
SD_SPDIF_OUT_BITSTREAM	Output the data that was input for the Core0 input block as a bit stream.
SD_SPDIF_OUT_BYPASS	Output the data that was input for the Core0 input block bypassing the internal SPU.
SD_SPDIF_COPY_NORMAL	Normal copy protection (first-generation recordable).

SD_SPDIF_COPY_PROHIBIT	Digital recording prohibited.

Note that a core cannot be specified for SD_C_SPDIF_MODE. Whichever core the settings are applied to, they will become general SPU2 settings.

Note that in a multithreaded environment, if this function is called simultaneously from more than one thread, the actual value that is set will be unpredictable.

Syntax for specifying entry:

For SD_C_SPDIF_MODE: SD_C_SPDIF_MODE (core specification is ignored)

Other than SD_C_SPDIF_MODE: SD_CORE_? | SD_C_*

In some cases, not setting anything for the SPDIF setting will not cause a problem during operation. However, for the purpose of complying with the standard, be sure to set the SPDIF setting properly.

(Example)

Set DVD for media, PCM for output, and prohibited for digital sound

```
sceSdSetCoreAttr( SD_C_SPDIF_MODE,
SD_SPDIF_MEDIA_DVD|SD_SPDIF_OUT_PCM|SD_SPDIF_COPY_PROHIBIT );
```

Return value

None

sceSdSetEffectAttr

Set the effect attribute

Library	*Introduced*	*Documentation last modified*
libsd	1.1	October 11, 2001

Syntax (IOP)

int sceSdSetEffectAttr (

int *core,*	Specifies the core. (0 or 1)
sceSdEffectAttr **attr***)**	Pointer to effect attribute structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

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

Description

Sets the effect attributes.

See the description of sceSdEffectAttr for more information.

Note that in a multithreaded environment, if this function is called simultaneously from more than one thread, the actual value that is set will be unpredictable.

Before executing this API, it is necessary to set the end address of the effect area (which is set using the SD_A_EEA macro). The starting address (ESA) is set within the API, according to the type of effect.

Return value

Status (unimplemented)

sceSdSetParam

Set register wrapper basic parameter

Library	*Introduced*	*Documentation last modified*
libsd	1.1	October 11, 2001

Syntax

void sceSdSetParam (	
u_short *register,*	Number of register in which the parameter will be set
u_short *value)*	Parameter value to be set in the register

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

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

Description

Sets the 16-bit parameter in the basic parameter registers and the volume registers.

Use this API for the SD_P_* and SD_VP_* series registers.

Note that if this function is called simultaneously in a multithreaded environment so that the same register is set from more than one thread, the actual value that is set will be unpredictable.

Syntax for specifying the register number:

For SD_P_* : SD_CORE_? | SD_P_*
For SD_VP_*: SD_CORE_? | SD_VOICE_?? | SD_VP_*

SD_VP_ENVX, SD_VP_VOLXL, SD_VP_VOLXR, and SD_P_MVOLX are read only, so they cannot be set.

Return value

None

sceSdSetSpu2IntrHandler

Set SPU2 interrupt handler

Library	*Introduced*	*Documentation last modified*
libsd	1.6	October 11, 2001

Syntax (IOP)

sceSdSpu2IntrHandler
sceSdSetSpu2IntrHandler (

sceSdSpu2IntrHandler *func,*	Pointer to interrupt handler
	Specifying NULL invalidates the interrupt handler.
void **data);*	Data address passed to interrupt handler func

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

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

Description

Sets the SPU2 interrupt handler.

This function should be called only if SPU interrupts have not been enabled.

Notes

Since interrupt handlers are executed independently from threads, a number of special issues must be considered when programming. For detailed information, refer to the warnings in \overview\iopkernl.

Return value

Pointer to interrupt handler that had been set previously

sceSdSetSwitch

Set register wrapper voice control parameter

Library	*Introduced*	*Documentation last modified*
libsd	1.1	October 11, 2001

Syntax

void sceSdSetSwitch (

u_short *register,*	Number of register in which parameter will be set
u_int *value)*	Parameter value (bit mask) set in register

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

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

Description

Sets the on/off flag for each voice in the voice control parameter register.

Use this API for the SD_S_* series registers.

Note that if this function is called simultaneously in a multithreaded environment so that the same register is set from more than one thread, the actual value that is set will be unpredictable.

<Syntax for specifying the register number>

For SD_S_* : SD_CORE_? | SD_S_*

SD_S_ENDX is read only, so it cannot be set.

Return value

None

sceSdSetTransIntrHandler

Set transfer interrupt handler

Library	*Introduced*	*Documentation last modified*
libsd	1.6	October 11, 2001

Syntax (IOP)

sceSdTransIntrHandler
sceSdSetTransIntrHandler (

int *channel,*	Transfer channel. 0 or 1 can be specified.
sceSdTransIntrHandler *func,*	Pointer to interrupt handler Specifying NULL invalidates the interrupt handler.
void **data);*	Data address passed to interrupt handler func

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

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

Description

Sets the transfer interrupt handler (excluding voice I/O transfers).

This function should be called only when a transfer is not being performed.

The timing when the interrupt begins varies according to the transfer mode setting. For SD_BLOCK_ONESHOT, the interrupt begins when a transfer of the specified size ends. For SD_BLOCK_LOOP, the interrupt begins at the midpoint and endpoint of the transfer size.

The interrupt begins not when a transfer of the specified size is performed, but when it is actually transferred to the I/O block within the SPU2.

Notes

Since interrupt handlers are executed independently from threads, a number of special issues must be considered when programming. For detailed information, please refer to the warnings in \overview\iopkernl.

Return Value

Pointer to interrupt handler that had been set previously

See Also

sceSdBlockTrans()

sceSdVoiceTrans

Transfer to SPU2 local memory

Library	*Introduced*	*Documentation last modified*
libsd	1.1	October 11, 2001

Syntax

int sceSdVoiceTrans (

short *channel,*	Transfer channel. 0 or 1 can be specified.
u_short *mode,*	Transfer mode
u_char *m_*addr,*	IOP memory-side address
u_int *s_*addr,*	SPU memory-side address
u_int *size*)	Transfer 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-enabled state)

Description

Performs transfers between SPU2 local memory (voice memory) and IOP memory.

Due to hardware restrictions, the SPU memory address must be a multiple of 16. If other values are specified, fractions are ignored.

Transfers are performed in 64-byte units. Note that, even if the transfer size is not a multiple of 64 bytes, the transfer will still be performed in 64-byte units.

Always use sceSdVoiceTransStatus() to confirm that the transfer has completed, except when the transfer end interrupt handler has been set with sceSdSetTransIntrHandler(). Another transfer cannot be started unless the end of the current transfer has been confirmed.

Values of bit mask for mode:

- Transfer direction
  SD_TRANS_MODE_WRITE 0
  SD_TRANS_MODE_READ 1
- Transfer device
  SD_TRANS_BY_DMA (0x0<<3)
  SD_TRANS_BY_IO (0x1<<3) (write only)

Notes

When the data transfer direction is SPU2 local memory => IOP memory during a USB isochronous transfer, the operation on the USB side will timeout and cannot be performed properly.

Return value

Number of bytes transferred. Negative value in the event of an error.

sceSdVoiceTransStatus

Get status of voice transfer

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

Syntax

u_int sceSdVoiceTransStatus (

short *channel,*	Transfer channel. 0 or 1 can be specified.
short *flag***)**	Operation flag SD_TRANS_STATUS_WAIT: Wait until the transfer has completed. SD_TRANS_STATUS_CHECK: Return the current state without waiting.

Calling conditions

flag= SD_TRANS_STATUS_WAIT	Can be called from a thread Multithread safe
flag=SD_TRANS_STATUS_CHECK	Can be called from an interrupt handler Can be called from a thread Multithread safe

Description

Get the status of voice transfer.

Based on the flag settings, blocking or non-blocking processes can be selected.

Return value

1: transfer complete; 0: transfer in progress.

When the flag in the interrupt context is set to SD_TRANS_STATUS_WAIT, a negative value is returned and an abormal termination occurs.

Callback Functions

sceSdSpu2IntrHandler

SPU2 interrupt handler

Library	*Introduced*	*Documentation last modified*
libsd	1.6	July 24, 2000

Syntax

typedef int (*sceSdSpu2IntrHandler)(

int *core_bit,*	Bits representing the core corresponding to the generated SPU2 interrupt
void **data)*	Data address registered using sceSdSetSpu2IntrHandler()

Description

This function is executed within an SPU2 IRQ interrupt. At that time, the value that represents the core for which the interrupt was generated as bits (only the low-order two bits are valid) and the address of data that was specified during registration are passed as arguments.

<core_bit>

Table 9-6

bit1	bit0	
0	1	SPU2 IRQ interrupt generated in CORE0
1	0	SPU2 IRQ interrupt generated in CORE1
1	1	SPU2 IRQ interrupts generated at the same time in both CORE0 and CORE1

Return value

Currently unused, always returns 0.

See also

sceSdSetSpu2IntrHandler()

sceSdTransIntrHandler

Transfer interrupt handler specification

Library	*Introduced*	*Documentation last modified*
libsd	1.6	July 24, 2000

Syntax

typedef int (*sceSdTransIntrHandler)(

int *channel,*	Transfer channel (0 or 1) specified when setting the handler using sceSdSetTransIntrHandler()
void **data)*	User data address specified when the handler was set using sceSdSetTransIntrHandler()

Description

This function is executed within the interrupt that is generated when a DMA transfer ends. At that time, the transfer channel number for which the interrupt was generated and the data address that was specified during registration are passed as arguments.

Return value

Currently unused, always returns 0.

See Also

sceSdSetTransIntrHandler()

Register Macros

SD_A_EEA

End address of working area for effects processing

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

Syntax

u_int sceSdGetAddr(SD_CORE_?ISD_A_EEA); //Get

void sceSdSetAddr(SD_CORE_?ISD_A_EEA, u_int value); //Set

Description

This register is used for digital effects processing. It specifies the end address of the working area.

Bit 17 must be set to 1, so only a 128-KB boundary can be specified.

Table 9-7

Bit	Symbol	Contents
0-22	ADDR	End address of work area for effects processing. Bits 0-16 should all be 1.

SD_A_ESA

Top address of working area for effects processing

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

Syntax

u_int sceSdGetAddr(SD_CORE_?lSD_A_ESA); //Get

void sceSdSetAddr(SD_CORE_?lSD_A_ESA, u_int value); //Set

Description

This register is used for digital effects processing. It specifies the top address of the working area.

Table 9-8

Bit	Symbol	Contents
0-22	ADDR	Top address of working area for effects processing

Note: When sceSdSetEffectAttr() of the effect-setting API is used, setting is performed within the API, so it is unnecessary to directly set S_A_ESA.

SD_A_IRQA

Set SPU2 Interrupt address

Library	*Introduced*	*Documentation last modified*
libsd	1.1	October 6, 2000

Syntax

u_int sceSdGetAddr(SD_CORE_?lSD_A_IRQA; //Get
void sceSdSetAddr(SD_CORE_?lSD_A_IRQA, u_int value); //Set

Description

This register specifies the address in local memory which, when accessed by each core, will cause an interrupt to the host (IOP).

Table 9-9

Bit	Symbol	Contents
0-22	ADDR	Address that causes the interrupt. Bits 0-3 should be 0.

SD_A_TSA

Transfer start address

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

Syntax

u_int sceSdGetAddr(SD_CORE_?|SD_A_TSA); //Get

void sceSdSetAddr(SD_CORE_?|SD_A_TSA, u_int value); //Set

Description

This register specifies the top address of local memory, which is used as the transfer destination for transfers to SPU2 local memory (except for transfers to the I/O block).

The value is immutable regardless of the execution state of the transfer.

When the value is changed during the transfer, both the operation and the transferred data will become uncertain.

Normally, the transfer start address is set within the library so it is not necessary to be set by the user.

Table 9-10

Bit	Symbol	Contents
0-22	ADDR	Top address of transfer area Bits 0-3 should be 0.

SD_P_AVOLL

Core external input volume (left)

SD_P_AVOLR

Core external input volume (right)

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

Syntax

u_short sceSdGetParam(SD_CORE_?lSD_P_AVOLx); //Get

void sceSdSetParam(SD_CORE_?lSD_P_AVOLx, u_short value); //Set

Description

These registers specify the volume of the core external input.

Table 9-11

Bit	Symbol	Contents
15-0	VALUE	Volume value

SD_P_BVOLL
Sound data input volume (left)

SD_P_BVOLR
Sound data input volume (right)

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

Syntax

u_short sceSdGetParam(SD_CORE_?lSD_P_BVOLx); //Get

void sceSdSetParam(SD_CORE_?lSD_P_BVOLx, u_short value); //Set

Description

These registers specify the volume of the sound data input.

Table 9-12

Bit	Symbol	Contents
15-0	VALUE	Volume value

SD_P_EVOLL
Effect return volume (left)

SD_P_EVOLR
Effect return volume (right)

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

Syntax

u_short sceSdGetParam(SD_CORE_?lSD_P_EVOLx); //Get

void sceSdSetParam(SD_CORE_?lSD_P_EVOLx, u_short value); //Set

Description

These registers specify the effect return volume.

Table 9-13

Bit	Symbol	Contents
15-0	VALUE	Volume value

SD_P_MMIX

Output type after voice mixing

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

Syntax

u_short sceSdGetParam(SD_CORE_?ISD_P_MMIX); //Get

void sceSdSetParam(SD_CORE_?ISD_P_MMIX, u_short value); //Set

Description

This register specifies the current output type (either normal or effect) as shown below.

Table 9-14

Bit	Symbol	Contents
11	MSNDL	Voice output (dry: L) -> Normal output
10	MSNDR	Voice output (dry: R) -> Normal output
09	MSNDEL	Voice output (wet: L) -> Effect output
08	MSNDER	Voice output (wet: R) -> Effect output
07	MINL	Sound data input (L) -> Normal output
06	MINR	Sound data input (R) -> Normal output
05	MINEL	Sound data input (L) -> Effect output
04	MINER	Sound data input (R) -> Effect output
03	SINL	Core external input (L) -> Normal output
02	SINR	Core external input (R) -> Normal output
01	SINEL	Core external input (L) -> Effect output
00	SINER	Core external input (R) -> Effect output

SD_P_MVOLL
Set master volume (left)

SD_P_MVOLR
Set master volume (right)

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

Syntax

u_short sceSdGetParam(SD_CORE_?lSD_P_MVOLx); //Get
void sceSdSetParam(SD_CORE_?lSD_P_MVOLx, u_short value); //Set

Description

These registers specify the master volume of each core.

The contents of the id field (bits 15-12) vary in value as shown below.

Table 9-15

ID	Meaning
0xxx	Fixed value specification mode The value is specified in bits 0-14. For a negative number, invert the phase.
1000	Linear increase mode (normal phase) Adds the value specified in 1Ts. Increases linearly to +1.0. The value is specified in bits 0-7. The current value should be positive.
1001	Linear increase mode (inverse phase) Adds the value specified in 1Ts. Decreases linearly to -1.0. The value is specified in bits 0-7. The current value should be negative.
1010	Linear decrease mode (normal phase) Adds the value specified in 1Ts. Decreases linearly to 0.0. The value is specified in bits 0-7. The current value should be positive.
1011	Linear decrease mode (inverse phase) Adds the value specified in 1Ts. Increases linearly to 0.0. The value is specified in bits 0-7. The current value should be negative.
1100	Pseudo inverse-exponential increase mode (normal phase) Adds in proportion to the value specified in 1Ts. Increases in a broken line to 1.0. The value is specified in bits 0-7. The current value should be positive.

ID	Meaning
1101	Pseudo inverse-exponential increase mode (inverse phase)
	Adds in proportion to the value specified in 1Ts.
	Increases in a broken line to -1.0.
	The value is specified in bits 0-7.
	The current value should be negative.
1110	Exponential decrease mode
	Multiplies by the value specified in 1Ts.
	The value is specified in bits 0-7.

SD_P_MVOLXL
Current master volume (left)

SD_P_MVOLXR
Current master volume (right)

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

Syntax

u_short sceSdGetParam(SD_CORE_?ISD_P_MVOLXx); //Get

Description

These registers specify the current master volume.

Read only. Cannot be set.

When MVOL is not in fixed value specification mode, the value changes every 1 Ts according to the volume change.

Table 9-16

Bit	Symbol	Contents
15-0	VALUE	Current value of volume

SD_S_ENDX

Endpoint reached flag

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

Syntax

u_int sceSdGetSwitch(SD_CORE_?ISD_S_ENDX); //Get

void sceSdSetSwitch(SD_CORE_?ISD_S_ENDX, u_int value); //Set

Description

This register indicates whether or not the endpoint block has been reached during sound generation processing for each voice. Read only. Cannot be set.

Table 9-17

Bit	Symbol	Contents
0	VOICE	Endpoint reached flag for voice 0
		0: Not reached
		1: Reached
…		
23	VOICE	Endpoint reached flag for voice 23
		0: Not reached
		1: Reached

By specifying key on, the bit which corresponds to that voice will become 0.

Also, by writing an arbitrary value (any number other than zero) to this register, all bits are cleared to 0.

SD_S_KOFF

Key off (end sound generation)

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

Syntax

u_int sceSdGetSwitch(SD_CORE_?lSD_S_KOFF); //Get

void sceSdSetSwitch(SD_CORE_?lSD_S_KOFF, u_int value); //Set

Description

This register specifies the value of key off (end of sound generation) for each voice. Sound generation will be ended for each voice when the corresponding bit is set to 1. After the state changes to key off, the envelope will transition to release. Sound will not necessarily switch off immediately.

Table 9-18

Bit	Symbol	Contents
0	VOICE	Switch key off for voice 0
…		
23	VOICE	Switch key off for voice 23

An interval of at least 2 Ts is required when continuously writing to the same register. When continuously writing with less than 2 Ts, the voice performing the actual end sound generation process is uncertain.

SD_S_KON

Key on (start sound generation)

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

Syntax

u_int sceSdGetSwitch(SD_CORE_?|SD_S_KON); //Get

void sceSdSetSwitch(SD_CORE_?|SD_S_KON, u_int value); //Set

Description

This register specifies the value of key on (i.e., start of sound generation) for each voice. Sound generation will be started for each voice when the corresponding bit is set to 1. Note that if a bit set to zero, it will not result in key off.

Table 9-19

Bit	Symbol	Contents
0	VOICE	Switch key on for voice 0
…		
23	VOICE	Switch key on for voice 23

The value read by this register is not reflected in the voice actually generated.

An interval of at least 2 Ts is required when continuously writing to the same register. When continuously writing with less than 2 Ts, the voice performing the actual end sound generation process is uncertain.

Also, key on can be specified by writing bit 1 again without specifying key off to the voice performing the sound generation process.

SD_S_NON

Noise generator assignment

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

Syntax

u_int sceSdGetSwitch(SD_CORE_?lSD_S_NON); //Get

void sceSdSetSwitch(SD_CORE_?lSD_S_NON, u_int value); //Set

Description

This register specifies whether or not a noise generator is assigned as the sound source for each voice.

Table 9-20

Bit	Symbol	Contents
0	VOICE	Specifies the sound source for voice 0. 0: OFF 1: ON
…		
23	VOICE	Specifies the sound source for voice 23. 0: OFF 1: ON

SD_S_PMON

Pitch modulation.

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

Syntax

u_int sceSdGetSwitch(SD_CORE_?ISD_S_PMON); //Get

void sceSdSetSwitch(SD_CORE_?ISD_S_PMON, u_int value); //Set

Description

This register specifies whether or not to apply pitch modulation to each voice.

The amplitude of the voice wave that is one less than that of the specified voice is used for modulation. Bit 0, corresponding to Voice0 cannot be specified.

Table 9-21

Bit	Symbol	Contents
1	VOICE	Specifies the pitch modulation of voice 1. 0: OFF 1: ON
…		
23	VOICE	Specifies the pitch modulation of voice 23. 0: OFF 1: ON

SD_S_VMIXL
Voice output mixing (dry left)

SD_S_VMIXR
Voice output mixing (dry right)

SD_S_VMIXEL
Voice output mixing (wet left)

SD_S_VMIXER
Voice output mixing (wet right)

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

Syntax

u_int sceSdGetSwitch(SD_CORE_?lSD_S_VMIXx); //Get

void sceSdSetSwitch(SD_CORE_?lSD_S_VMIXx, u_int value); //Set

Description

These registers specify whether or not the output of each voice is output to dry left / dry right / wet left / wet right.

Dry means the no-effect side, and wet means the effect side.

Table 9-22

Bit	Symbol	Contents
0	VOICE	Output switch for voice 0
		0: Not output to the relevant channel
		1: Output to the relevant channel
…		
23	VOICE	Voice 23 endpoint reached flag
		0: Not output to the relevant channel
		1: Output to the relevant channel

SD_VA_LSAX

Loop point address

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

Syntax

u_int sceSdGetAddr(SD_CORE_?|SD_VOICE_?|SD_VA_LSAX); //Get

void sceSdSetAddr(SD_CORE_?|SD_VOICE_?|SD_VA_LSAX,
u_int value); //Set

Description

This register indicates the top address of the block which is specified at the loop point in the waveform data. It is initially set after reaching the loop point block.

During sound generation (after 4 Ts have passed following key on. Rewriting is ignored if less than 4 Ts) this register's value can be changed. However, in such cases, the address set to the corresponding voice takes precedence and the loop point block information is invalidated until the next key on for the voice.

Table 9-23

Bit	Symbol	Contents
0-22	ADDR	Address of loop point Bits 0-3 should be 0.

SD_VA_NAX

Address of waveform data that should be read next

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

Syntax

u_int sceSdGetAddr(SD_CORE_?ISD_VOICE_?ISD_VA_NAX); //Get

Description

This register indicates the waveform data address to be read next, in the waveform data. It is updated automatically as sound generation proceeds.

Read only. Cannot be set.

Table 9-24

Bit	Symbol	Contents
0-22	ADDR	Address of waveform data to be read next

SD_VA_SSA

Top address of waveform data

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

Syntax

u_int sceSdGetAddr(SD_CORE_?ISD_VOICE_?ISD_VA_SSA); //Get

void sceSdSetAddr(SD_CORE_?ISD_VOICE_?ISD_VA_SSA,
u_int value); //Set

Description

This register specifies the top address of the waveform data, which will be used as the sound source for each voice.

Table 9-25

Bit	Symbol	Contents
0-22	ADDR	Top address of waveform data Bits 0-3 should be 0.

SD_VP_ADSR1

Envelope

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

Syntax

u_short sceSdGetParam(SD_CORE_?lSD_VOICE_?lSD_VP_ADSR1); //Get

void sceSdSetParam(SD_CORE_?lSD_VOICE_?lSD_VP_ADSR1, //Set
u_short value);

Description

This register specifies the envelope for each voice as shown below.

Table 9-26

Bit	Symbol	Contents
15	AM	Attack rate mode
		0: Linear increase
		1: Pseudo exponential increase
14-8	AR	Attack rate
7-4	DR	Decay rate
3-0	SL	Sustain level

SD_VP_ADSR2

Envelope (2)

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

Syntax

u_short sceSdGetParam(SD_CORE_?ISD_VOICE_?ISD_VP_ADSR2); //Get

void sceSdSetParam(SD_CORE_?ISD_VOICE_?ISD_VP_ADSR2, //Set
u_short value);

Description

This register specifies the envelope for each voice as shown below.

Table 9-27

Bit	Symbol	Contents
15-13	SM	Sustain rate mode
		000: Linear increase mode
		010: Linear decrease mode
		100: Pseudo-exponential increase mode
		110: Exponential decrease mode
12-6	SR	Sustain rate
5	RM	Release rate mode
		0: Linear decrease mode
		1: Exponential decrease mode
4-0	RR	Release rate

SD_VP_ENVX

Current value of envelope

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

Syntax

u_short sceSdGetParam(SD_CORE_?ISD_VOICE_?ISD_VP_ENVX); //Get

Description

This register specifies the current value of envelope for each voice.

Read only. Cannot be set.

When the specification of SR and RR of the envelope is a linear decrease specification, sometimes only 1 Ts is negative. Also, when generating sound in non-loop wave pattern data, ENVX becomes 0, regardless of the envelope state, at the point when the bit corresponding to that voice in the ENDX register became 1.

Table 9-28

Bit	Symbol	Contents
15-0	VALUE	Current value of envelope

SD_VP_PITCH

Sound generation pitch

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

Syntax

u_short sceSdGetParam(SD_CORE_?ISD_VOICE_?ISD_VP_PITCH); //Get

void sceSdSetParam(SD_CORE_?ISD_VOICE_?ISD_VP_PITCH, //Set
u_short value);

Description

This register specifies the sound generation pitch for each voice.

Table 9-29

Bit	Symbol	Contents
15-0	VALUE	Specified pitch value

If the pitch of the fundamental tone is f0, the relationship between the specified pitch value (VALUE) and the generated pitch f is as follows:

```
f = VALUE * f0 / 4096
```

In addition, if the sound source is used as a noise generator, there will be no change in auditory sensation when the specified pitch is varied. The noise pitch is specified using a separate API.

The pitch specification affects the performance of sound generation. When the specified pitch value is low, sound generation takes longer.

SD_VP_VOLL
Voice volume (left)

SD_VP_VOLR
Voice volume (right)

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

Syntax

u_short sceSdGetParam(SD_CORE_?ISD_VOICE_?ISD_VP_VOLx); //Get

void sceSdSetParam(SD_CORE_?ISD_VOICE_?ISD_VP_VOLx, //Set
u_short value);

Description

These registers specify the volume mode for each voice.

The contents of the id field (bits 15-12) vary in value as shown below.

Table 9-30

ID	Meaning
0xxx	Fixed value specification mode The value is specified in bits 0-14. For a negative number, invert the phase.
1000	Linear increase mode (normal phase1) Adds the value specified in 1Ts. Increases linearly to +1.0. The value is specified in bits 0-7. The current value should be positive.
1001	Linear increase mode (inverse phase) Adds the value specified in 1Ts. Decreases linearly to -1.0. The value is specified in bits 0-7. The current value should be negative.
1010	Linear decrease mode (normal phase) Adds the value specified in 1Ts. Decreases linearly to 0.0. The value is specified in bits 0-7. The current value should be positive.
1011	Linear decrease mode (inverse phase) Adds the value specified in 1Ts. Increases linearly to 0.0. The value is specified in bits 0-7. The current value should be negative.
1100	Pseudo inverse-exponential increase mode (normal phase) Adds in proportion to the value specified in 1Ts. Increases in a broken line to 1.0. The value is specified in bits 0-7. The current value should be positive.

ID	Meaning
1101	Pseudo inverse-exponential increase mode (inverse phase) Increases in proportion to the value specified in 1Ts. Increases in a broken line to -1.0. Specify the value in bit 0-7. The current value should be negative.
1110	Exponential decrease mode Multiplies by the value specified in 1Ts. Specify the value in bit 0-7.

SD_VP_VOLXL
Current volume (left)

SD_VP_VOLXR
Current volume (right)

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

Syntax

u_short sceSdGetParam(SD_CORE_?lSD_VOICE_?lSD_VP_VOLXx); //Get

Description

These registers specify the current volume for each voice.

Read only. Cannot be set.

When VOL is not in fixed value specification mode, the value changes every 1 Ts according to the volume change.

Table 9-31

Bit	Symbol	Contents
15-0	VALUE	Current volume value

Chapter 10: CSL SE Sequencer
Table of Contents

Structures

sceSESqEnv

SE Sequence Environment

Library	*Introduced*	*Documentation last modified*
modsesq	2.1	January 4, 2001

Structure

typedef struct {

unsigned int *songNum;*	SE SongChunk number that is currently being performed or has been selected (Currently, invalid)
unsigned char *masterVolume;*	Input port/master volume
char *masterPanpot;*	Input port/master panpot
unsigned short *masterTimeScale;*	Input port/master timescale
unsigned int *status;*	Input port performance state
int *defaultOutPort;*	Default output port number where SE sequence set is output
sceSESqPortAssignment *outPort* **[sceSESqNumSeqStream];**	Output port number where SE sequence set is output setNo: SE sequence set number port: Output port number
unsigned char *system* **[sceSESqEnvSize];**	Module's input variable area

} sceSESqEnv;

Description

This is an environment buffer for managing the state of the performance and attributes for each input port.

The values of the members are initialized as necessary by calling sceSESq_Init() and sceSESq_Load(). For details, please see the description of each function.

sceSESqPortAssignment

SE sequence Set output port specification information

Library	*Introduced*	*Documentation last modified*
modsesq	2.1	January 4, 2001

Structure

typedef struct {	
unsigned char *setNo;*	SE sequence set number
unsigned char *port;*	Output port number
} sceSESqPortAssignment;	

Description

This structure is used for specifying the target output port of an SE sequence.

All SE sequences that are contained in the SE sequence set having setNo as the SE sequence set number are output to the output port specified by port.

Functions

sceSESq_ATick

Periodic data conversion processing

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

Syntax

int sceSESq_ATick(

sceCslCtx **module_context)*	Address of Module Context

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 is periodically called after a certain time interval.

Each time this function is called, the SE sequence to be performed proceeds by converting a portion of it over the time interval specified by the interval argument in the sceSESq_Init() function, to an SE stream.

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

Return value

Always sceSESqNoError

sceSESq_GetEnv

Get environment address

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

Syntax

sceSESqEnv *sceSESq_GetEnv(	
sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number)*	Input port number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function gets the environment address that corresponds to the port number.

Return value

Environment address

sceSESq_Init

Initialize

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

Syntax

int sceSESq_Init(	
sceCslCtx **module_context,*	Address of Module Context
unsigned int *interval)*	ATick() calling interval expressed 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-enabled state)

Description

This function initializes the internal environment of the SE sequencer module and sets initial values for the environment.

The members of sceSESqEnv that are initialized by this function and their values are shown below.

Table 10-1

Member	Value
defaultOutPort	sceSESqEnv_NoOutPortAssignment
outPort [].setNo	sceSESqEnv_NoSeqSet;
outPort [].port	sceSESqEnv_NoOutPortAssignment

The members of the sceCslSeStream output buffer that are initialized by this function and their values are shown below.

Table 10-2

Member	Value
validsize	0

Return value

sceSESqNoError	Normal termination
sceSESqError	Error in environment or arguments

sceSESq_Load

Load sequence data

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

Syntax

int sceSESq_Load(	
sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number)*	Input port number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function registers SQ sequence data in the specified input port.

If performance is in progress for the specified input port, operation will not be guaranteed if that input port's environment attributes have changed, or if sceSESq_Load() is called.

The members of sceSeSqEnv that are initialized by this function and their values are shown below.

Table 10-3

Member	Value
songNum	sceSESqEnv_NoSeSongNum (Currently invalid)
masterVolume	sceSESq_Volume0db
masterPanpot	sceSESq_PanpotCenter
masterTimeScale	sceSESq_BaseTimeScale
status	0
defaultOutPort	sceSESqEnv_NoOutPortAssignment
outPort [].setNo	sceSESqEnv_NoSeqSet;
outPort [].port	sceSESqEnv_NoOutPortAssignment

Return value

sceSESqNoError	Normal termination
sceSESqError	Error in environment or arguments

sceSESq_SelectSeq

Assign SE Sequence

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

Syntax

int sceSESq_SelectSeq(

sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Input port number
unsigned char *set_number,*	SE Sequence Set number
unsigned char *seq_number)*	SE Sequence 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 assigns a sequence ID to an SE Sequence that is to be performed.

Subsequently, the sequence ID can be used when performing processing for that SE Sequence.

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

Return value

Non-negative (>=0)	Sequence ID
sceSESqError	Error in environment or arguments

sceSESq_SeqGetStatus

Get SE sequence state

Library	*Introduced*	*Documentation last modified*
modsesq	2.1	July 2, 2001

Syntax

int sceSESq_SeqGetStatus(	
sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Input port number
unsigned int *sesq_id)*	Sequence ID

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function returns the state of the SE sequence for the specified sequence ID.

Return value

>=0 The returned SE sequence state, defined by the bit OR of the following values

sceSESqStat_ready	Can be performed
sceSESqStat_inPlay	Performance is in progress
sceSESqStat_dataEnd	End of data was reached
sceSESqStat_seqIDAutoUnselect	After the performance ends, cancel the sequence ID assignment automatically
sceSESqError	Error in environment or arguments

sceSESq_SeqIsDataEnd

Get SE sequence state (is end ofdata?)

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

Syntax

Bool sceSESq_SeqIsDataEnd(

sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Input port number
unsigned int *sesq_id)*	Sequence ID

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function checks whether or not the SE sequence of the specified sequence ID has reached the end of data.

Return value

True	Reached the end of the data
False	Did not reach the end of the data

sceSESq_SeqIsInPlay

Get SE sequence state (is performing?)

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

Syntax

Bool sceSESq_SeqIsInPlay(	
sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Input port number
unsigned int *sesq_id)*	Sequence ID

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function checks whether or not the SE sequence of the specified sequence ID is being performed.

Return value

True	Performance is in progress
False	Performance is not in progress

sceSESq_SeqPlaySwitch

Start or stop performance of SE sequence

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

Syntax

int sceSESq_SeqPlaySwitch(

sceCslCtx **module_context,*	Address of Module Context	
unsigned int *port_number,*	Input port number	
int *sesq_id,*	Sequence ID	
int *command)*	Performance command	
	sceSESq_SeqPlayStop	Stop performance
	sceSESq_SeqPlayStart	Start performance
	sceSESq_SeqPlayTerminate	Explicitly terminate the performance
	sceSESq_SeqPlayStart	Control operation by Oring in the bit shown below
	sceSESq_SeqPlaySeqIDAutoUnselect	Automatically cancel the sequence ID assignment after the performance ends

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 starts/stops a performance.

When the performance start is specified with sceSESq_SeqPlayStart and at the same time, sceSESq_SeqPlaySeqIDAutoUnselect is ORed into command, unselect processing for the specified sequence ID will be automatically performed internally to cancel the assignment when the performance of the SE sequence reaches the end of data or when the performance is explicitly terminated with sceSESq_SeqPlayStop/Terminate.

If this function is called with *command* set to sceSESq_SeqPlayStop, and if the performance of the SE sequence had already reached the end of the data and stopped, then no processing is performed.

To mute any voices which are still producing sound after the performance has ended, call this function with command set to sceSESq_SeqPlayTerminate.

Also, to mute any voices which are still producing sound after the sequence ID assignment was canceled with sceSESq_UnselectSeq() or when sceSESq_SeqPlaySeqIDAutoUnselect was specified when the performance was started, set the SE message ID for the SE sequence with sceSESq_SeqSetSEMsgID() before the performance is started and call sceSESq_SeqTerminateVoice().

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

Return value

sceSESqNoError	Normal termination
sceSESqError	Error in environment or arguments

sceSESq_SeqSetSEMsgID

Specify SE message ID that SE sequence will use

Library	*Introduced*	*Documentation last modified*
modsesq	2.3	July 2, 2001

Syntax

int sceSESq_SeqSetSEMsgID(	
sceCslCtx **module_context,*	Module Context address
unsigned int *port_number,*	Input port number
int *sesq_id,*	Sequence ID
unsigned int *se_message_id***)**	SE message ID

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function sets the SE message ID (4 bytes) that the SE sequence specified by the sequence ID sesq_id will use.

This function can be used any time after the sequence ID has been assigned with sceSESq_SelectSeq() until the performance is started with sceSESq_SeqPlaySwitch(). If the sequence ID assignment is canceled with sceSESq_UnselectSeq(), the SE message ID that was specified by this function will also be disabled for the relevant sequence ID.

Return value

sceSESqNoError	Normal termination
sceSESqError	Environment or argument is invalid

sceSESq_SeqTerminateVoice

Explicitly terminate voices for which sound was produced by SE sequence

Library	*Introduced*	*Documentation last modified*
modsesq	2.3	October 11, 2001

Syntax

int sceSESq_SeqTerminateVoice (	
sceCslCtx **module_context,*	Module Context address
unsigned int *in_port_number,*	Input port number
unsigned int *out_port_number,*	Output port number
int *se_message_id,*	SE message ID
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 explicitly terminates voices for which sound was produced by an SE sequence.

The specified se_message_id is ANDed with the specified mask and compared with the AND of the specified mask and the SE message IDs kept by the voices. All voices where the result is the same will be explicitly terminated.

This function should only be used to mute voices which are still producing sound after the assignment of the sequence ID (sesq_id) is canceled with sceSESq_UnselectSeq().

When a sequence ID is reused, sceSESqSeqSetSEMsgID() should be used in advance to set the SE message ID that a module will use for that sequence ID.

Once a sequence ID assignment is valid, set the performance command to sceSESq_SeqPlayStop or sceSESq_SeqPlayTerminate when calling sceSESq_SeqPlaySwitch().

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

Return value

sceSESqNoError	Normal termination
sceSESqError	Environment or argument is invalid

sceSESq_UnselectSeq

Cancel assignment of Sequence ID

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

Syntax

int sceSESq_UnselectSeq(	
sceCslCtx **module_context,*	Address of Module Context
unsigned int *port_number,*	Input port number
unsigned char *sesq_id)*	Sequence ID

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 cancels the assignment of the specified sequence ID.

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

Return value

sceSESqNoError	Normal termination
sceSESqError	Error in environment or arguments