libsndp Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

Datatypes

SceSsSndpCtx

sndp environment

Definition

```
#include <libsndp.h>
typedef struct SceSsSndpCtx{
        SceUInt32 system[SCE_SNDP_SNDP_CTX_SIZE/sizeof(SceUInt32)];
} SceSsSndpCtx;
```

Members

system	Used internally by libsndp.

Description

This structure should be provided for each set of sound data (PHD, PBD).

See Also

`sceSsBindSoundData()`

SceSsSmfCtx

smf environment

Definition

```
#include <libsndp.h>
typedef struct{
        SceUInt32 system[SCE_SNDP_SMF_CTX_SIZE/sizeof(SceUInt32)];
} SceSsSmfCtx;
```

Members

system	Used internally by libsndp.

Description

This structure should be provided for each SMF to be used.

See Also

`sceSsSMFBind()`

SceSsKeyOnParam

Key on parameters

Definition

```
#include <libsndp.h>
typedef struct{
        unsigned char vel;
        unsigned char pan;
        unsigned char sendVel;
        unsigned char sendPan;
        int addPitch;
} SceSsKeyOnParam;
```

Members

vel	Velocity (0 - 127)
pan	Panpot (0 - 64 - 127)
sendVel	Effect send velocity (0 - 127)
sendPan	Effect send panpot (0 - 64 - 127)
addPitch	Added pitch

Description

This structure is used for setting the velocity and added pitch when starting the production of sound specified by key on and note on.

The values of the members are modified based on the tone parameter settings.

If the results when the values are modified exceed the upper or lower limit for the respective values, the result is clamped to the upper or lower limit, respectively.

For the panpot or effect send panpot, 0 corresponds to the leftmost position, 64 to the middle, and 127 to the rightmost position.

See Also

`sceSsNoteOnByTone(),sceSsKeyOnByTone(),sceSsVoiceNoteOnByTone()`

`sceSsVoiceKeyOnByTone(),sceSsNoteOn()`

SceSsEffectParam

Effect parameters

Definition

```
#include <libsndp.h>
typedef struct{
        unsigned int setMask;
        signed int type;
        unsigned int delayTime;
        unsigned int feedBack;
        unsigned int effectVolumeLeft;
        unsigned int effectVolumeRight;
        unsigned int drySwitch;
        unsigned int wetSwitch;
} SceSsEffectParam;
```

Members

setMask	Bitwise OR of the desired parameters that are set (see below).
type	Effect type
delayTime	Delay time (0 - 127)
feedBack	Feedback (0 - 127)
effectVolumeLeft	Left channel (Lch) effect volume (0 - 4096)
effectVolumeRight	Right channel (Rch) effect volume (0 - 4096)
drySwitch	Dry-side sound ON/OFF switch (`SCE_SNDP_OFF` = OFF, `SCE_SNDP_ON` = ON)
wetSwitch	Wet-side (sound with effect applied) sound ON/OFF switch (`SCE_SNDP_OFF` = OFF, `SCE_SNDP_ON` = ON)

Description

To *setMask*, specify the bitwise OR of the following parameter types.

Value	Description
SCE_SNDP_FX_PARAM_ALL	Set all parameters
SCE_SNDP_FX_PARAM_TYPE	Set the effect type (set the specified value for *type*)
SCE_SNDP_FX_PARAM_PARAM	Set the effect parameters (set the specified values for *delayTime* and *feedBack*)
SCE_SNDP_FX_PARAM_EFFECTVOLUME	Set the effect volume (set the specified values for *effectVolumeLeft* and *effectVolumeRight*)
SCE_SNDP_FX_PARAM_SWITCH	Set the effect switches (set the specified values for *drySwitch* and *wetSwitch*)

type specifies the effect type. The following constants are provided.

```
SCE_SNDP_FX_TYPE_OFF
SCE_SNDP_FX_TYPE_ROOM
SCE_SNDP_FX_TYPE_STUDIOA
SCE_SNDP_FX_TYPE_STUDIOB
SCE_SNDP_FX_TYPE_STUDIOC
SCE_SNDP_FX_TYPE_HALL
SCE_SNDP_FX_TYPE_SPACE
SCE_SNDP_FX_TYPE_ECHO
```

```
SCE_SNDP_FX_TYPE_DELAY
SCE_SNDP_FX_TYPE_PIPE
```

`delayTime` specifies the delay time.
`feedBack` specifies the amount of feedback.
`effectVolumeLeft` and *`effectVolumeRight`* specify the left and right effect volumes.
`drySwitch` specifies the dry-side switch, and *`wetSwitch`* specifies the wet-side switch.
For details about effect settings, refer to the "SAS Overview" and "SAS Reference" documents.

Initialization / Termination Functions

sceSsGetNeededMemorySize

Get memory size (in bytes) required for initialization

Definition

```
#include <libsndp.h>
SceInt32 sceSsGetNeededMemorySize(
        const char *config,
        SceSize *outSize
);
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Not multithread safe.

Arguments

config	For future expansion (currently specify an empty string "".)
outSize	Pointer to variable for storing memory size required for initialization

Return Values

When the function completes normally, then `SCE_OK` is returned.

When an error occurs, a negative value (< 0) is returned.

(For details, see the List of Error Codes.)

Description

This function obtains the memory size required for libsndp initialization.

This function internally calls `sceSasGetNeededMemorySize()` of SAS.

Be sure to allocate the size obtained with this function for the memory size required for libsndp initialization.

Examples

```
#include <libsndp.h>

int main(void)
{
        SceSize bufferSize;
        void *buffer;
        SceInt32 result;

        result = sceSsGetNeededMemorySize("", &bufferSize);
        buffer = malloc(bufferSize);
        result = sceSsInit(32, "", buffer, bufferSize);

        return 0;
}
```

sceSsInit

Initialization processing

Definition

```
#include <libsndp.h>
SceInt32 sceSsInit(
        SceUInt32 maxVoice,
        const char *config,
        void *buffer,
        SceSize bufferSize
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

maxVoice	Number of voices to be used (1- 32)
config	For future expansion (currently specify pointer to an empty string)
buffer	Pointer to Initialization buffer
bufferSize	Initialization buffer size

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes in this document and the "SAS Reference" document.)

Description

This function initializes libsndp's voice management system, the SMF sequencer, and SAS.

To use libsndp, this function or `sceSsInitWithGrain()` must be called first.

Since this function performs SAS initialization, when you use libsndp, do not try to initialize SAS by calling the SAS initialization function.

Calling this function sets the time for one unit of granularity to 256 samples.

Be sure to allocate the size obtained with `sceSsGetNeededMemorySize()` for the memory size required for libsndp initialization.

Example

```
#include <libsndp.h>

int main(void)
{
        unsigned int useVoice = 32; /* Number of voices to use */
        void* buffer;
        SceSize bufferSize;
        void *buffer;
        int result;

        result = sceSsGetNeededMemorySize("", &bufferSize);
        buffer = malloc(bufferSize);
        result = sceSsInit(useVoice, "", buffer, bufferSize);
        return 0;
}
```

sceSsInitWithGrain

Initialization processing (with granularity specification)

Definition

```
#include <libsndp.h>
SceInt32 sceSsInitWithGrain(
        SceUInt32 maxVoice,
        const char *config,
        SceUInt32 grain,
        void *buffer,
        SceSize bufferSize
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

maxVoice	Number of voices to use (1 - 32)
config	For future expansion (currently specify NULL)
grain	Granularity (valid range is 64 to 2048 in multiples of 32; i.e., 64, 96, 128, ..., 2016, 2048)
buffer	Pointer to Initialization buffer
bufferSize	Initialization buffer size

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it means that an error occurred.

(For details, see the List of Error Codes in this document and the "SAS Reference" document.)

Description

This function initializes the voice management system within libsndp, the SMF sequencer, and SAS.

To use libsndp, this function or `sceSsInit()` must be called first.

Since this function also performs SAS initialization, do not use a SAS function to initialize SAS if you are also using libsndp.

As the value of *grain* decreases, sound production changes can be expressed in a shorter granularity time, but the CPU processing load will increase.

Example

```
#include <libsndp.h>

int main(void)
{
        SceInt32   result;
        unsigned int   useVoice = 32; /* Number of voices to use */
        unsigned int   Grain = 1024; /* Granularity */
        void* buffer;
        SceSize bufferSize;
        void *buffer;

        result = sceSsGetNeededMemorySize("", &bufferSize);
        buffer = malloc(bufferSize);
        result = sceSsInitWithGrain(useVoice, "", Grain,
                                          buffer, bufferSize);
        return 0;
}
```

sceSsExit

Termination processing

Definition

```
#include <libsndp.h>
SceInt32 sceSsExit(
        void **outBuffer,
        SceSize *outBufferSize
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

outBuffer	Pointer to variable for storing pointer to memory specified by initialization Specify NULL when not used.
outBufferSize	Pointer to variable for storing memory size specified by initialization Specify NULL when not used.

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes in this document and the "SAS Reference" document.)

Description

This function performs libsndp termination processing and SAS termination processing.

Always execute this function when exiting an application that has been using libsndp.

Since this function performs SAS termination processing, do not execute the SAS termination function.

Example

```
#include <libsndp.h>

int main(void)
{
        SceSize bufferSize;
        void *buffer;
        SceInt32 result;

        result = sceSsGetNeededMemorySize("", &bufferSize);
        buffer = malloc(bufferSize);
        result = sceSsInit(32, "", buffer, bufferSize);

        result = sceSsExit(&buffer, &bufferSize);
        free(buffer);
        return 0;
}
```

Sound Functions

sceSsSynthesis

libsndp periodic processing

Definition

```
#include <libsndp.h>
SceInt32 sceSsSynthesis(
        short *pOut
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Multithread safe for key on, key off, note on, and note off functions

Arguments

pOut	Pointer to waveform output buffer

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

Calling this function causes SMF sequence processing and voice processing to be performed for one unit of granularity. Note that this function also calls `sceSasCore()` internally. After processing is completed, the voice states are updated to prepare for voice processing until the next time this function is called.

Since `sceSasCore()` is called internally by this function, do not call `sceSasCoreWithMix()` or `sceSasCore()`.

Operations that cause a change in voice state (such as note on, note off, key on, and key off) can be performed by another thread while this function is executing.

Example

```
#include <libsndp.h>

int main(void)
{
        int   result;
        short *pOut;

        result = sceSsSynthesis(pOut);

        return 0;
}
```

sceSsSynthesisWithMix

libsndp periodic processing (with external PCM mixing function)

Definition

```
#include <libsndp.h>
SceInt32 sceSsSynthesisWithMix(
        short *pInOut,
        SceInt32 lVol,
        SceInt32 rVol
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Multithread safe for key on, key off, note on, and note off functions

Arguments

pInOut	Pointer to waveform output buffer
lVol	Volume value for left channel of external PCM input
rVol	Volume value for right channel of external PCM input

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it means that an error occurred.

(For details, see the List of Error Codes.)

Description

Calling this function causes SMF sequence processing and voice processing for one unit of granularity to be performed. This function also calls `sceSasCoreWithMix()` internally. Consequently, PCM data in a specified area is mixed to produce the resultant output. Then, the voice states are updated to prepare for voice processing until the next time this function is called.

pInOut must have 64-byte alignment.

During mixing, the volume of the input PCM can be changed. The following shows the valid range for the volume.

0 <= *lVol*, *rVol* <= `SCE_SNDP_VOLUME_MAX`

Since `sceSasCoreWithMix()` is called internally by this function, be sure not to use `sceSasCore()` or `sceSasCoreWithMix()`.

Operations that cause a change in voice state (such as note on, note off, key on, and key off) can be performed by another thread while this function is executing.

Example

```
#include <libsndp.h>

int main(void)
{
        int   result;
        short *pInOut;
        SceInt32 lVol = SCE_SNDP_VOLUME_MAX;
        SceInt32 rVol = SCE_SNDP_VOLUME_MAX;

        /*
              After lVol and rVol processing are performed for the sound in the
              pInOut buffer (such as background music), the sound is mixed with
              the sound generated by SAS.
        */
        result = sceSsSynthesisWithMix(pInOut,lVol,rVol);

        return 0;
}
```

sceSsVoiceSetReserveMode

Set a voice to be managed by libsndp

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceSetReserveMode(
        SceUInt32 voiceNum,
        SceUInt32 reserveMode
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

voiceNum	Voice number for which the management status is to be set	
reserveMode	Status to be set	
	`SCE_SNDP_ON`	Place voice under libsndp management
	`SCE_SNDP_OFF`	Remove voice from libsndp management

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function turns ON/OFF the management state of libsndp for voices reserved by `sceSsInit()`.

It returns an error if a voice is specified that has not been reserved by `sceSsInit()`.

The default voice management status is `SCE_SNDP_ON`.

libsndp will not use voices that were removed from management.

A voice that was removed from libsndp management can be used by libsas.

To set the management status of a voice, execute this function after stopping sound production for that voice.

Example

```
#include <libsndp.h>

int main(void)
{
        int   result;

        /* Remove voice 0 from libsndp management */
        result = sceSsVoiceSetReserveMode(0, SCE_SNDP_OFF);

        return 0;
}
```

sceSsBindSoundData

Register sound data

Definition

```
#include <libsndp.h>
SceInt32 sceSsBindSoundData(
        SceSsSndpCtx *sndpCtx,
        void *phd,
        void *pbd,
        SceUInt32 pbfs
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

sndpCtx	Pointer to libsndp environment structure
phd	Pointer to PHD data
pbd	Pointer to PBD data
pbfs	Output sampling frequency (44100 or 48000)

Return Value

The allocated phd ID is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function registers one set of sound data (*phd*, *pbd*) in libsndp.

Be sure to prepare an sndp environment structure for each set of sound data.

Sound data is accessed using the phd ID returned by this function.

Specify the output sampling frequency of libsndp in the *pbfs* argument. Specify 44100 or 48000.

Up to 128 sets of sound data can be registered.

Example

```
#include <libsndp.h>

int main(void)
{
        int phdID;
        SceSsSndpCtx sndpCtx;
        unsigned char *phd, *pbd;


        phdID = sceSsBindSoundData(&sndpCtx, phd, pbd, 48000);

        return 0;
}
```

Notes

The `sndpCtx` structure in the argument does not need to be initialized. Be sure to allocate memory and pass it as a work area.

For details about sound data (phd, pbd), refer to the document entitled, "PHD/PBD Formats."

If 48000 is specified in argument `pbfs` and PSP™ (PlayStation®Portable) data (created for playback at 44.1 KHz) is specified for `phd` and `pbd`, a pitch conversion that differs from that of the PSP™ environment may occur, creating a difference in sound quality.

See Also

`SceSsSndpCtx`

sceSsUnbindSoundData

Cancel registered sound data

Definition

```
#include <libsndp.h>
SceInt32 sceSsUnbindSoundData(
        SceUInt32 phdID
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

phdID	phd ID for which registered sound data is to be canceled

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function cancels sound data (phd, pbd) that has been registered.

After the sound data corresponding to the phd ID is canceled by this function, that sound data can no longer be used.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsUnbindSoundData(0);

        return 0;
}
```

See Also

`sceSsBindSoundData()`

sceSsNoteOnByTone

Note on by tone

Definition

```
#include <libsndp.h>
SceInt32 sceSsNoteOnByTone(
        SceUInt32 phdID,
        SceUInt32 toneIndex,
        SceUInt32 note,
        SceUInt32 keyOnID,
        SceSsKeyOnParam *keyOnParam
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

phdID	phd ID that contains tone for which note on is to be performed
toneIndex	Tone index for which note on is to be performed
note	Note number (0 - 127)
keyOnID	Key on ID
keyOnParam	Pointer to `SceSsKeyOnParam` structure

Return Value

The voice number for which note on was performed is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function produces sound using the pitch specified by *note* and based on the tone parameters within the *phd* indicated by *toneIndex* (index of structure `toneParamCtx` of tone attribute data). Although the voice to be used is allocated automatically, sound may not actually be produced, depending on the priority that was set in the tone parameters. (Please refer to the "PHD/PBD Formats" document for details on the format of tone data.)

Also, when the corresponding voice is paused, `SCE_SNDP_ERROR_PAUSEVOICE` is returned.

The actual note for which sound is produced is determined from the note number that is set in the argument and the center note in the tone parameter. For example, if note on is performed for the same note as the center note in the tone parameter, the pitch of the fundamental tone is used to generate sound for its waveform.

(The center detune and detune values are also taken into consideration for the actual pitch.)

Actual sound generation starts after `sceSsSynthesis()` is called. However, if sound generation is stopped or another sound is generated for the allocated voice before `sceSsSynthesis()` is called, no sound is actually generated.

The volume and added pitch are set in the `SceSsKeyOnParam` structure.

The key on ID is an identifier used for generating sound .

Note that since the key on IDs 0x80000000 to 0xFFFFFFFF are reserved by libsndp, they should not be used.

Example

```
#include <libsndp.h>

int main(void)
{
        int voiceNum;
        SceSsKeyOnParam keyOnParam;

        keyOnParam.vel = 127;
        keyOnParam.pan = 64;
        keyOnParam.sendVel = 127;
        keyOnParam.sendPan = 64;
        keyOnParam.addPitch = 0;

        /* Generate sound for phdID = 0, tone index = 0, note = 64, and */
        /* key on Id = 0 */
        voiceNum = sceSsNoteOnByTone(0, 0, 64, 0, &keyOnParam);

        return 0;
}
```

See Also

SceSsKeyOnParam

sceSsKeyOnByTone

Key on by tone

Definition

```
#include <libsndp.h>
SceInt32 sceSsKeyOnByTone(
        SceUInt32 phdID,
        SceUInt32 toneIndex,
        SceUInt32 pitch,
        SceUInt32 keyOnID,
        SceSsKeyOnParam *keyOnParam
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

phdID	phd ID that includes the tone for which key on is to be performed
toneIndex	Tone index for which key on is to be performed
pitch	Pitch (0x0001 - 0x4000)
keyOnID	Key on ID
keyOnParam	Pointer to `SceSsKeyOnParam` structure

Return Value

The voice number for which note on was performed is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function produces sound using the pitch specified by *pitch* and based on the tone parameters within the *phd* indicated by *toneIndex*. Although the voice to be used is allocated automatically, sound may not actually be produced, depending on the priority that was set in the tone parameters.

Also, when the corresponding voice is paused, `SCE_SNDP_ERROR_PAUSEVOICE` is returned.

The actual note for which sound is produced is determined from the note number that is set in the argument and the center note in the tone parameter. For example, if note on is performed for the same note as the center note in the tone parameter, the pitch of the fundamental tone is used to generate sound for its waveform.

(The center detune and detune values are also taken into consideration for the actual pitch.)

Actual sound generation starts after `sceSsSynthesis()` is called. However, if sound generation is stopped or another sound is generated for the allocated voice before `sceSsSynthesis()` is called, no sound is actually generated.

The volume and added pitch are set in the `SceSsKeyOnParam` structure.

The key on ID is an identifier used for generating sound .

Note that since the key on IDs 0x80000000 to 0xFFFFFFFF are reserved by libsndp, they should not be used.

Example

```
#include <libsndp.h>

int main(void)
{
        int voiceNum;
        SceSsKeyOnParam keyOnParam;

        keyOnParam.vel = 127;
        keyOnParam.pan = 64;
        keyOnParam.sendVel = 127;
        keyOnParam.sendPan = 64;
        keyOnParam.addPitch = 0;

        /* Generate sound for phdID = 0, tone index = 0, pitch = 0x1000, */
        /* and key on Id = 0 */
        voiceNum = sceSsKeyOnByTone(0, 0, 0x1000, 0, &keyOnParam);

        return 0;
}
```

See Also

SceSsKeyOnParam

sceSsVoiceNoteOnByTone

Note on by tone (specifying voice)

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceNoteOnByTone(
        SceUInt32 phdID,
        SceUInt32 voiceNum,
        SceUInt32 toneIndex,
        SceUInt32 note,
        SceUInt32 keyOnID,
        SceSsKeyOnParam *keyOnParam
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

phdID	phd ID that includes the tone for which note on is to be performed
voiceNum	Voice number for which note on is to be performed
toneIndex	Tone index for which note on is to be performed
note	Note number (0 - 127)
keyOnID	Key on ID
keyOnParam	Pointer to `SceSsKeyOnParam` structure

Return Value

The voice number for which note on was performed is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function produces sound using the pitch specified by *note* and based on the tone parameters within the *phd* indicated by *toneIndex*. Sound is forcibly produced by the specified voice.

The actual note for which sound is produced is determined from the note number that is set in the argument and the center note in the tone parameter. For example, if note on is performed for the same note as the center note in the tone parameter, the pitch of the fundamental tone is used to generate sound for its waveform.

(The center detune and detune values are also taken into consideration for the actual pitch.)

Actual sound generation starts after `sceSsSynthesis()` is called. However, if sound generation is stopped or another sound is generated for the voice that is set before `sceSsSynthesis()` is called, no sound is actually generated.

Also, when the corresponding voice is paused, `SCE_SNDP_ERROR_PAUSEVOICE` is returned.

The volume and added pitch are set in the `SceSsKeyOnParam` structure.

Voices that can be specified in the *voiceNum* argument can only be voices that were previously registered using `sceSsInit()`.

The key on ID is an identifier used for generating sound .

Note that since the key on IDs 0x80000000 to 0xFFFFFFFF are reserved by libsndp, they should not be used.

Example

```
#include <libsndp.h>

int main(void)
{
        int voiceNum;
        SceSsKeyOnParam keyOnParam;

        keyOnParam.vel = 127;
        keyOnParam.pan = 64;
        keyOnParam.sendVel = 127;
        keyOnParam.sendPan = 64;
        keyOnParam.addPitch = 0;

        /* Generate sound for phdID = 0, tone index = 0, note = 64, */
        /* and key on Id = 0 */
        voiceNum = sceSsVoiceNoteOnByTone(0, 0, 0, 64, 0, &keyOnParam);

        return 0;
}
```

See Also

SceSsKeyOnParam

sceSsVoiceKeyOnByTone

Key on by tone (voice specification)

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceKeyOnByTone(
        SceUInt32 phdID,
        SceUInt32 voiceNum,
        SceUInt32 toneIndex,
        SceUInt32 pitch,
        SceUInt32 keyOnID,
        SceSsKeyOnParam *keyOnParam
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

phdID	phd ID that includes the tone for which key on is to be performed
voiceNum	Voice number for which key on is to be performed
toneIndex	Tone index for which key on is to be performed
pitch	Pitch (0x0001 - 0x4000)
keyOnID	Key on ID
keyOnParam	Pointer to `SceSsKeyOnParam` structure

Return Value

The voice number for which key on was performed is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function produces sound using the pitch specified by *pitch* and based on the tone parameters within the *phd* indicated by *toneIndex*. Sound is forcibly produced by the specified voice.

The actual note for which sound is produced is determined from the note number that is set in the argument and the center note in the tone parameter. For example, if note on is performed for the same note as the center note in the tone parameter, the pitch of the fundamental tone is used to generate sound for its waveform.

(The center detune and detune values are also taken into consideration for the actual pitch.)

Actual sound generation starts after `sceSsSynthesis()` is called. However, if sound generation is stopped or another sound is generated for the voice that is set before `sceSsSynthesis()` is called, no sound is actually generated.

Also, when the corresponding voice is paused, `SCE_SNDP_ERROR_PAUSEVOICE` is returned.

The volume and added pitch are set in the `SceSsKeyOnParam` structure.

The key on ID is an identifier used for generating sound .

Note that since the key on IDs 0x80000000 to 0xFFFFFFFF are reserved by libsndp, they should not be used.

Example

```
#include <libsndp.h>

int main(void)
{
        int voiceNum;
        SceSsKeyOnParam keyOnParam;

        keyOnParam.vel = 127;
        keyOnParam.pan = 64;
        keyOnParam.sendVel = 127;
        keyOnParam.sendPan = 64;
        keyOnParam.addPitch = 0;

        /* Generate sound for phdID = 0, tone index = 0, pitch = 0x1000, */
        /* and key on Id = 0 */
        voiceNum = sceSsVoiceKeyOnByTone(0, 0, 0, 0x1000, 0, &keyOnParam);

        return 0;
}
```

See Also

SceSsKeyOnParam

sceSsVoiceSetSustainHold

Set sustain hold state for a voice

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceSetSustainHold(
        SceUInt32 voiceNum,
        SceUInt32 sustainHold
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

voiceNum	Voice number for which sustain hold state is to be set	
sustainHold	Sustain hold state to be set	
	`SCE_SNDP_OFF`	Sustain hold off
	`SCE_SNDP_ON`	Sustain hold on

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function sets the sustain state for a voice. If an operation is performed to stop sound generation for a voice that has sustain hold on, and if the sound being generated is a loop waveform, sound generation will not be stopped and the voice will transition to sustain state. To stop sound generation for a voice in sustain state, sustain hold must be switched off.

Also, when the corresponding voice is paused, `SCE_SNDP_ERROR_PAUSEVOICE` is returned.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;


        result = sceSsVoiceSetSustainHold(0, SCE_SNDP_ON);

        return 0;
}
```

sceSsVoiceKeyoff

Voice key off

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceKeyoff(
        SceUInt32 voiceNum
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

voiceNum	Voice for which key off is to be performed

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function performs key off for the specified voice. The voice will transition to sustain state when sustain hold is on. The actual key off is performed when sustain hold is turned off.

Also, when the corresponding voice is paused, `SCE_SNDP_ERROR_PAUSEVOICE` is returned.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        /* Perform key off for voice 0 */
        result = sceSsVoiceKeyoff(0);

        return 0;
}
```

sceSsVoiceSoundOff

Voice sound off

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceSoundOff(
        SceUInt32 voiceNum
);
```

Calling Conditions

Can be called from an interrupt handler
Can be called from a thread (does not depend on interrupt-disabled or -enabled state)
Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

voiceNum	Voice for which sound off is to be performed

Return Value

When the function completes normally, `SCE_OK` is returned.
If the return value is negative (< 0), it indicates an error.
(For details, see the List of Error Codes.)

Description

This function quickly ends sound production for the specified voice. It turns sustain hold off, performs key off, and sets the shortest possible ADSR release time. It is useful when you want to quickly end sound production for a voice while calling `sceSsSynthesis()` successively. Also, when the corresponding voice is paused, `SCE_SNDP_ERROR_PAUSEVOICE` is returned.

Example

```
This example performs sound off for the voice of the specified MIDI channel.

#include <libsndp.h>

/*
 Arguments
   midich : MIDI channel
   id : Key-on ID
 */
int main(int midich, int id)
{
        int result;
        int vnum;
        unsigned int vbit;

        /* Get bit pattern of voice which is generating sound for the specified
                  MIDI channel  */
        vbit = sceSsGetVoice(midich, id);

        for(vnum=0; (vnum<SCE_SAS_VOICE_MAX) || (vbit!=0); vnum++){
                if(vbit & 0x0001){ /* If ON voice */
                        /* Turn sound off for voice vnum */
                        result = sceSsVoiceSoundOff(vnum);
                        if(result < 0) return result; /* End if error occurs */
                        }
                vbit >>= 1;
          }
          return result;
 }
```

sceSsVoiceSetPitch

Add pitch to a voice producing sound

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceSetPitch(
        SceUInt32 voiceNum,
        SceInt32 addPitch
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

voiceNum	Target voice number
addPitch	Added pitch to be set

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function adds the pitch to the specified voice. If a negative value was set for the added pitch, the pitch is reduced. A negative number can also be specified for *addPitch*.

If the added pitch falls below 0x1 or exceeds 0x4000, it is clamped to 0x1 or 0x4000, respectively.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        /* Add pitch 0x200 to voice 0 */
        result = sceSsVoiceSetPitch(0, 0x200);

        return 0;
}
```

sceSsVoiceSetVelocity

Set voice velocity

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceSetVelocity(
        SceUInt32 voiceNum,
        SceUInt32 velocity
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

voiceNum	Target voice number
velocity	Velocity to be set (0 - 127)

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function sets the velocity of a voice. The velocity is a value that is relative to the volume that was set in the tone parameters in the PHD.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        /* Set velocity 127 for voice 0 */
        result = sceSsVoiceSetVelocity(0, 127);

        return 0;
}
```

sceSsVoiceSetPanpot

Set voice panpot

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceSetPanpot(
        SceUInt32 voiceNum,
        SceUInt32 panpot
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

voiceNum	Target voice number
panpot	Panpot to be set (0 - 64 - 127)

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function sets the panpot of a voice. The value 0 corresponds to the leftmost position, 127 to the rightmost position, and 64 to the middle.

This value is added to the actual panpot that was set for the tone parameters in the PHD. If the result falls below 0 or exceeds 127, it is clamped to 0 or 127, respectively.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        /* Set panpot 127 for voice 0 */
        result = sceSsVoiceSetPanpot(0, 127);

        return 0;
}
```

sceSsVoiceSetPitchBend

Set pitch bend of voice

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceSetPitchBend(
        SceUInt32 voiceNum,
        SceUInt32 bendValue
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

voiceNum	Voice number for which pitch bend is to be set
bendValue	Pitch bend value to be set 0 to 8191 (Bend Down) 8192 (Center) 8193 to 16383 (Bend Up)

Return Value

If processing completes normally, `SCE_OK` is returned.

If an error occurs, the return value is negative (< 0).

(For details, see the List of Error Codes.)

Description

This function sets the pitch bend of a voice.

The pitch varies within the range that was set by the bend range of the tone parameter in the PHD.

bendValue is a range that matches the MIDI pitch bend value.

bendValue and pitch vary linearly relative to the musical scale (cents).

The pitch bend effect is effective only for a voice for which sound is being produced according to a specified note (`sceSsNoteOnByTone()`, `sceSsVoiceNoteOnByTone()`, or `sceSsNoteOn()`).

Since the bend range width is ambiguous for a voice for which sound is produced by directly specifying the pitch, the pitch bend effect is not effective in those cases (it is not effective for `sceSsKeyOnByTone()` or `sceSsVoiceKeyOnByTone()`).

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        /* Set a maximum pitch bend of 16383 for voice 0 */
        result = sceSsVoiceSetPitchBend(0, 16383);

        return 0;
}
```

sceSsVoiceSetSendVelocity

Set effect send velocity of a voice

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceSetSendVelocity(
        SceUInt32 voiceNum,
        SceUInt32 sendVelocity
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

voiceNum	Voice number that is set
sendVelocity	Effect send velocity that is set (0 - 127)

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function sets the velocity of a voice. The velocity is a value that is relative to the volume that was set in the tone parameters in the PHD.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        /* Set effect send velocity 127 for voice 0 */
        result = sceSsVoiceSetSendVelocity(0, 127);

        return 0;
}
```

sceSsVoiceSetSendPanpot

Set effect send panpot of a voice

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceSetSendPanpot(
        SceUInt32 voiceNum,
        SceUInt32 sendPanpot
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

voiceNum	Target voice number
sendPanpot	Effect send panpot to be set (0 - 64 - 127)

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function sets the effect send panpot of a voice. The value 0 corresponds to the leftmost position, 127 to the rightmost position, and 64 to the middle.

This value is added to the actual effect send panpot that was set for the tone parameters in the PHD. If the result falls below 0 or exceeds 127, it is clamped to 0 or 127, respectively.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        /* Set effect send panpot 127 for voice 0 */
        result = sceSsVoiceSetSendPanpot(0, 127);

        return 0;
}
```

sceSsVoiceAllKeyOff

All key off

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceAllKeyOff(
        void
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

None

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

If voices are included that have errors, the error code of the last voice that caused an error is returned.

(For details, see the List of Error Codes.)

Description

This function keys off all voices that are being managed by libsndp (performs key off and puts ADSR in release state).

Sound production is stopped after turning sustain hold state off for any voice for which sustain hold has been set.

Also, when the corresponding voice is paused, `SCE_SNDP_ERROR_PAUSEVOICE` is returned.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        /* Stop sound production of all voices */
        result = sceSsVoiceAllKeyOff();

        return 0;
}
```

sceSsVoiceAllSoundOff

All sound off

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceAllSoundOff(
        void
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

None

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

If voices are included that have errors, the error code of the last voice that caused an error is returned.

(For details, see the List of Error Codes.)

Description

This function quickly ends sound production for all voices under libsndp management. It turns sustain hold off, performs key off and sets the shortest possible ADSR release time. It is useful when you want to quickly end sound production for all voices while calling `sceSsSynthesis()` successively.

When `sceSsVoiceAllKeyOff()` is used, the ADSR release time of the voices which were keyed off is long and it will take time before the voices stop producing sound. By comparison, the voices will take less time to finish when `sceSsVoiceAllSoundOff()` is used. However, note that reverberation will not stop. Also, when any one of the voices is paused, `SCE_SNDP_ERROR_PAUSEVOICE` is returned.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        /* Stop sound production for all voices */
        result = sceSsVoiceAllSoundOff();

        return 0;
}
```

sceSsVoiceSetPause

Pause voice playback

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceSetPause(
        SceUInt32 voiceNum,
        SceUInt32 mode
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

voiceNum	Voice number to be paused or for which pause is to be canceled (0 to 31)
mode	Pause state (1: Set pause; 0: Cancel pause)

Return Value

If the function completes normally, `SCE_OK` is returned.

When an error occurs, a negative value (< 0) is returned.

(For details, see the List of Error Codes.)

Description

This function sets or cancels pause state for the voice having the specified number.

If `sceSsVoiceGetStatus()` is executed for the voice during a pause, the value in effect immediately before the pause is returned.

If a reserved pause is specified, an error (`SCE_SNDP_ERROR_RESERVEDVOICE`) is returned.

If a note on or note off is performed for a paused voice, it will be ignored by the voice. An error (`SCE_SNDP_ERROR_PAUSEVOICE`) may be returned, depending on the function.

sceSsVoiceGetEndFlag

Get sound production end flag for a voice

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceGetEndFlag(
        SceUInt32 voiceNum
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe

Arguments

voiceNum	Voice number for which end flag is to be obtained

Return Value

The sound production end flag status is returned.

0: The end of sound production has not been reached

1: The end of sound production has been reached

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function gets the sound production end flag for a voice.

If envelope processing has ended and sound production has ended, the end flag is 1. If sound is being produced, the end flag is 0. All end flags will be set to 1 immediately after initialization (`sceSsInit()`).

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        /* Get the sound production end flag status of voice 0 */
        result = sceSsVoiceGetEndFlag(0);

        return 0;
}
```

sceSsVoiceGetEnvelope

Get envelope value of a voice

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceGetEnvelope(
        SceUInt32 voiceNum
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe

Arguments

voiceNum	Voice number for which envelope value is to be obtained

Return Value

Envelope value

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes in this document and the "SAS Reference" document.)

Description

Returns the envelope peak value of the specified voice.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        /* Get envelope value of voice 0 */
        result = sceSsVoiceGetEnvelope(0);

        return 0;
}
```

sceSsVoiceGetStatus

Get voice state

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceGetStatus(
        SceUInt32 voiceNum
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe

Arguments

voiceNum	Voice number for which voice state is to be obtained

Return Value

The state of the specified voice is returned.

```
SCE_SNDP_VOICESTATUS_RESERVED
SCE_SNDP_VOICESTATUS_IDLE
SCE_SNDP_VOICESTATUS_BUSY
SCE_SNDP_VOICESTATUS_RELEASE
SCE_SNDP_VOICESTATUS_SUSTAIN
```

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function gets the state of a voice.

`SCE_SNDP_VOICESTATUS_RESERVED` corresponds to a voice that was removed from libsndp management.

(To remove a voice from libsndp management, use `sceSsVoiceSetReserveMode()`.)

`SCE_SNDP_VOICESTATUS_IDLE` corresponds to a voice that is first used as a result of a note on event that immediately follows libsndp initialization, or a voice that has finished vocalization.

`SCE_SNDP_VOICESTATUS_RELEASE` corresponds to a voice when ADSR is being released by a vocalization stop (note off) event, or a voice when ADSR is released by the next `sceSsSynthesis()` call.

`SCE_SNDP_VOICESTATUS_SUSTAIN` corresponds to a voice for which the vocalization stop (note off) event occurred but hold is being sustained by the `sceSsVoiceSetSustainHold()` function.

`SCE_SNDP_VOICESTATUS_BUSY` corresponds to a voice that is vocalized by the next `sceSsSynthesis()` call or a voice for which vocalization is being sustained by a vocalization start (note on) event.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsVoiceGetStatus(0);

        return 0;
}
```

sceSsVoiceGetLevel

Get voice level setting

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceGetLevel(
        SceUInt32 voiceNum,
        SceInt32 *lVol,
        SceInt32 *rVol
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe

Arguments

voiceNum	Voice number for which level is to be obtained
lVol	Pointer for returning the left volume value
rVol	Pointer for returning the right volume value

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it means that an error occurred.

(For details, see the List of Error Codes in this document and the "SAS Reference" document.)

Description

This function gets the volume of a voice.

Specifically, it gets the setting of the `sceSasSetVolume()` function that is called internally by libsndp.

The range of volume values is as follows.

-SCE_SNDP_VOLUME_MAX <= *lVol*, *rVol* <= SCE_SNDP_VOLUME_MAX

A negative value inverts the phase.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;
        int nvoice;
        int VolumeL, VolumeR;

        /* Get the voice envelope value */
        nvoice = 0;  /*  Voice 0  */
        result = sceSsVoiceGetLevel(nvoice, &VolumeL, &VolumeR);
        return 0;
}
```

sceSsVoiceGetSendLevel

Get effect send level setting of voice

Definition

```
#include <libsndp.h>
SceInt32 sceSsVoiceGetSendLevel(
        SceUInt32 voiceNum,
        SceInt32 *lVol,
        SceInt32 *rVol
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe

Arguments

voiceNum	Voice number for which level is to be obtained
lVol	Pointer for returning the left effect send volume value
rVol	Pointer for returning the right effect send volume value

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it means that an error occurred.

(For details, see the List of Error Codes in this document and the "SAS Reference" document.)

Description

This function gets the effect send volume of a voice.

Specifically it gets the setting of the `sceSasSetVolume()` function that is called internally by libsndp.

The range of volume values is as follows.

-`SCE_SNDP_VOLUME_MAX` <= *Vol*, *rVol* <= `SCE_SNDP_VOLUME_MAX`

A negative value inverts the phase.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;
        int nvoice;
        int VolumeL, VolumeR;

        /* Get the voice envelope value */
        nvoice = 0;  /*  Voice 0  */
        result = sceSsVoiceGetSendLevel(nvoice, &VolumeL, &VolumeR);
        return 0;
}
```

sceSsKeyOffByID

Key off a voice producing sound by its key on ID

Definition

```
#include <libsndp.h>
SceUInt32 sceSsKeyOffByID(
        SceUInt32 keyOnID
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

keyOnID	Key on ID for which key off is to be performed

Return Value

The bit pattern of the voice for which key off was performed is returned.

If the return value is zero, it means key off was not performed for any voice.

The relationship between the bit No. and voice No. is shown below.

voice31	voice30		voice1	voice0
bit31	bit30	...	bit1	bit0

Description

This function performs key off for a voice that is producing sound corresponding to key on ID.

However, a voice for which the sustain hold state was set to on switches to sustain state, and key off is performed when the sustain hold state is turned off.

Also, when the corresponding voices include a paused voice, the operation for the paused voice is ignored.

Example

```
#include <libsndp.h>

int main(void)
{
        unsigned int result;

        /* Stop sound production of voice with key on ID = 0 */
        result = sceSsKeyOffByID(0);

        return 0;
}
```

sceSsGetVoice

Get sound-producing voice using MIDI channel and key on ID

Definition

```
#include <libsndp.h>
SceUInt32 sceSsGetVoice(
        SceUInt32 midiChannel,
        SceUInt32 keyOnID
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe

Arguments

midiChannel	MIDI channel for which voice is to be obtained (0 to 15 = MIDI channels 1 to 16)
keyOnID	Key on ID for which voice is to be obtained

Return Value

The bit pattern of the voice that was obtained is returned.

If the return value is zero, it means that no voice was obtained.

The relationship between the bit No. and voice No. is shown below.

voice31	voice30		voice1	voice0
bit31	bit30	...	bit1	bit0

Description

This function gets the voice producing sound from its MIDI channel and key on ID.

Example

```
#include <libsndp.h>

int main(void)
{
        unsigned int result;

        /* Get voice with key on MIDI channel = 1 and key on ID = 0 */
        result = sceSsGetVoice(0, 0);

        return 0;
}
```

sceSsGetPause

Get pause states of voices

Definition

```
#include <libsndp.h>
SceUInt32 sceSsGetPause(
        void
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe

Arguments

None

Return Value

A bit pattern indicating the pause states of the voices that were obtained is returned.

A 1 indicates that the voice is paused, and a 0 indicates that the pause was canceled.

The relationship between the bit No. and voice No. is as follows.

voice31	voice30		voice1	voice0
bit31	bit30	···	bit1	bit0

Description

This function gets the pause state of each voice.

sceSsNoteOn

Note on

Definition

```
#include <libsndp.h>
SceUInt32 sceSsNoteOn(
        SceUInt32 phdID,
        SceUInt32 midiChannel,
        SceUInt32 midiProgram,
        SceUInt32 midiNote,
        SceUInt32 sustain,
        SceSsKeyOnParam *keyOnParam,
        SceUInt32 keyOnID
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

phdID	phd ID that contains the program for which note on is to be performed
midiChannel	MIDI channel for which note on is to be performed (0 to 15 = MIDI channels 1 to 16)
midiProgram	MIDI program number for which note on is to be performed (0 - 127)
midiNote	Note number for which note on is to be performed (0 - 127)
sustain	Sustain hold state when note on is performed `SCE_SNDP_ON` Sustain hold on `SCE_SNDP_OFF` Sustain hold off
keyOnParam	Pointer to `SceSsKeyOnParam` structure
keyOnID	Key on ID

Return Value

The bit pattern of the voice for which note on was performed is returned.

If the return value is zero, it means that sound was not produced for any voice.

The relationship between the bit No. and voice No. is shown below.

voice31	voice30		voice1	voice0
bit31	bit30	···	bit1	bit0

Description

This function performs note on for *`midiChannel`* using the tone parameters corresponding to the MIDI program number and note number. If there are multiple corresponding MIDI program numbers and note numbers, sound is produced for all the matching entries.

If no relevant program exists or if the note is outside of the tone parameter range, sound is not produced. Also, when the corresponding voices include a paused voice, the operation for the paused voice is ignored.

The actual note for which sound is produced is determined from the note number that is set in the argument and the center note in the tone parameter. For example, if note on is performed for the same note as the center note in the tone parameter, the pitch of the fundamental tone is used to generate sound for its waveform.

(The center detune and detune values are also taken into consideration for the actual pitch.)

Actual sound generation starts after `sceSsSynthesis()` is called. However, if sound generation is stopped or another sound is generated for the voice that generated sound before `sceSsSynthesis()` is called, no sound is actually generated.

The volume and added pitch are set in the `SceSsKeyOnParam` structure.

The key on ID is an identifier used for generating sound .

Note that since the key on IDs 0x80000000 to 0xFFFFFFFF are reserved by libsndp, they should not be used.

Example

```
#include <libsndp.h>

int main(void)
{
        unsigned int result;
        SceSsKeyOnParam keyOnParam;

        keyOnParam.vel = 127;
        keyOnParam.pan = 64;
        keyOnParam.sendVel = 127;
        keyOnParam.sendPan = 64;
        keyOnParam.addPitch = 0;

        /* Generate sound using phdID = 0, MIDI channel = 1, */
        /* program number = 0, note = 64, sustain state = off, and key on Id
= 0 */
        result= sceSsNoteOn(0, 0, 0, 64, SCE_SNDP_OFF, &keyOnParam, 0);

        return 0;
}
```

See Also

`SceSsKeyOnParam`

sceSsNoteOff

Note off

Definition

```
#include <libsndp.h>
SceUInt32 sceSsNoteOff(
        SceUInt32 midiChannel,
        SceUInt32 midiNote,
        SceUInt32 keyOnID
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

midiChannel	MIDI channel for which note off is to be performed (0 to 15 = MIDI channels 1 to 16)
midiNote	Note number for which note off is to be performed (0 - 127)
keyOnID	Key on ID for which note off is to be performed

Return Value

The bit pattern of the voice for which note off was performed is returned.

If the return value is zero, note off is not performed for any voice.

Description

This function performs note off for the sound-producing voice corresponding to *midiChannel*, *midiNote*, and *keyOnID*.

However, if the voice has its sustain hold state set to on, it will transition to sustain state and note off will be performed when the sustain hold state is turned off.

Also, when the corresponding voices include a paused voice, the operation for the paused voice is ignored

Example

```
#include <libsndp.h>

int main(void)
{
        unsigned int result;

        /* Stop the sound-producing voice with MIDI channel = 1, note = 64, */
        /* and key on ID = 0 */
        result= sceSsNoteOff(0, 64, 0);

        return 0;
}
```

sceSsSetSustainHold

Set sustain hold state of voice by specifying MIDI channel and key on ID

Definition

```
#include <libsndp.h>
SceUInt32 sceSsSetSustainHold(
        SceUInt32 midiChannel,
        SceUInt32 sustainHold,
        SceUInt32 keyOnID
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe for the `sceSsSynthesisWithMix()` and `sceSsSynthesis()` functions

Arguments

midiChannel	MIDI channel for which sustain hold state is to be set (0 to 15 = MIDI channels 1 to 16)	
sustainHold	Sustain hold state	
	`SCE_SNDP_ON`	Sustain hold on
	`SCE_SNDP_OFF`	Sustain hold off
keyOnID	Key on ID for which sustain hold state is to be set	

Return Value

The bit pattern of the voice for which sustain hold state was set is returned.

If the return value is zero, it means that sustain hold state was not set for any voice.

Description

This function sets the sustain state for a voice with the specified MIDI channel and key on ID. If an operation is performed to stop sound generation for a voice that has sustain hold on, and if the sound being generated is a loop waveform, sound generation will not be stopped and the voice will transition to sustain state. To stop sound generation for a voice in sustain state, sustain hold must be switched off.

Also, when the corresponding voices include a paused voice, the operation for the paused voice is ignored.

Example

```
#include <libsndp.h>

int main(void)
{
        unsigned int result;

        /* Set sustain state to on for the voice with MIDI channel = 1 and */
        /* key on ID = 0 */
        result= sceSsSetSustainHold(0, SCE_SNDP_ON, 0);
```

```
                return 0;
        }
```

Effect Function

sceSsSetEffectParam

Set effect contents

Definition

```
#include <libsndp.h>
SceInt32 sceSsSetEffectParam(
        SceSsEffectParam *effectParam
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

effectParam	Pointer to `SceSsEffectParam` structure

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes in this document and the "SAS Reference" document.)

Description

This function sets parameters for the effects you wish to perform in the members of the `SceSsEffectParam` structure.

The bitwise OR of the types of parameters to be set is specified for *setMask*, which is a member of the `SceSsEffectParam` structure. For example, if `SCE_SNDP_FX_PARAM_TYPE` is specified for *setMask*, the *type* parameter, which is a member of the `SceSsEffectParam` structure, is set.

Also, if `SCE_SNDP_FX_PARAM_ALL` is specified for *setMask*, all parameters are set.

For details about the settings, refer to the "SAS Overview" and "SAS Reference" documents.

The following parameters can be specified for *type*.

```
SCE_SNDP_FX_TYPE_OFF
SCE_SNDP_FX_TYPE_ROOM
SCE_SNDP_FX_TYPE_STUDIOA
SCE_SNDP_FX_TYPE_STUDIOB
SCE_SNDP_FX_TYPE_STUDIOC
SCE_SNDP_FX_TYPE_HALL
SCE_SNDP_FX_TYPE_SPACE
SCE_SNDP_FX_TYPE_ECHO
SCE_SNDP_FX_TYPE_DELAY
SCE_SNDP_FX_TYPE_PIPE
```

Example

```
#include <libsndp.h>

int main(void)
{
        int result;
        SceSsEffectParam effectParam;

        /* Set SCE_SNDP_FX_TYPE_STUDIOC for the effect type,
        4096 for the left and right effect volume, and set the effect
        switch on for both the dry and wet sides */

        effectParam.setMask = SCE_SNDP_FX_PARAM_TYPE |
                     SCE_SNDP_FX_PARAM_EFFECTVOLUME |
                     SCE_SNDP_FX_PARAM_SWITCH;
        effectParam.type = SCE_SNDP_FX_TYPE_STUDIOC;
        effectParam.effectVolumeLeft = 4096;
        effectParam.effectVolumeRight = 4096;
        effectParam.drySwitch = SCE_SNDP_ON;
        effectParam.wetSwitch = SCE_SNDP_ON;

        result= sceSsSetEffectParam(&effectParam);

        return 0;
}
```

See Also

SceSsEffectParam

Service Functions

sceSsNote2Pitch

Convert note value to pitch value

Definition

```
#include <libsndp.h>
SceUShort16 sceSsNote2Pitch(
        SceUShort16 centerNote,
        SceShort16 centerFine,
        SceUShort16 note,
        SceShort16 fine
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe

Arguments

centerNote	Base note during sampling
centerFine	Fine for the base note during sampling (semitone is 128)
note	Note number
fine	Fine for note (semitone is 128)

Return Value

The pitch is returned.

Description

This function calculates the pitch from the center note and sound-producing note.

Since the return value may exceed 0x4000, verify that the upper bound is not exceeded.

Example

```
#include <libsndp.h>

int main(void)
{
        unsigned short pitch;

        pitch= sceSsNote2Pitch(64, 0, 64, 0);

        return 0;
}
```

sceSsPitch2Note

Convert pitch to note

Definition

```
#include <libsndp.h>
SceUShort16 sceSsPitch2Note(
        SceUShort16 centerNote,
        SceShort16 centerFine,
        SceUShort16 pitch
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe

Arguments

centerNote	Base note during sampling
centerFine	Fine for base note during sampling
pitch	Pitch

Return Value

The note value (upper 8 bits are the note and lower 8 bits are the fine) is returned.

Description

This function calculates the sound-producing note from the center note and sound-producing pitch.

Example

```
#include <libsndp.h>

int main(void)
{
        unsigned short note;

        note = sceSsPitch2Note(64, 0, 64, 0x1000);

        return 0;
}
```

SMF Functions

sceSsSMFBind

Register SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFBind(
        SceSsSmfCtx *smfCtx,
        void *smf,
        SceUInt32 phdID,
        SceUInt32 tbfs
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

smfCtx	Pointer to smf environment structure
smf	Pointer to smf data
phdID	phd ID for which binding is to be performed
tbfs	Output sampling frequency (44100 or 48000)

Return Value

The smf ID that was assigned is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

An `SceSsSmfCtx` must be prepared for each set of data.

Note that separate `SceSsSmfCtx` structures must be prepared for different playback environments, even if the data is the same.

The *smfCtx* structure in the arguments does not need to be initialized. Be sure to allocate memory for it and pass it as a work area.

Specify the output sampling frequency of libsndp in the *tbfs* argument. Specify 44100 or 48000.

This function associates an SMF (Standard MIDI Format) with a PHD and returns the smf ID.

Up to 127 associations can be made.

If registration fails, it means that there may be a problem with the SMF format.

The SMF that can be used is only type 0.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;
        SceSsSmfCtx smfCtx;
        char *smf;

        result = sceSsSMFBind(&smfCtx, smf, 0, 48000);

        return 0;
}
```

Notes

When the tempo is 60, the SMF processing resolution is 1/172 for a quarter note. As a result, MIDI events that are generated with a lower resolution are quantized according to this resolution.

If a large number of MIDI events occur during the same unit of granularity, playback may get delayed. Therefore, either disperse the event timings or reduce the number of MIDI events to the minimum that are required.

Argument `tbfs` is used to generate the playback tempo of SMF.

See Also

`SceSsSmfCtx`

sceSsSMFUnbind

Cancel SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFUnbind(
        SceUInt32 smfID
);
```

Calling Conditions

Can be called from an interrupt handler
Can be called from a thread (does not depend on interrupt-enabled or disabled state)
Not multithread safe

Arguments

smfID	smfID to cancel

Return Value

The smfID that was assigned is returned.
If the return value is negative (< 0), an error is returned.
(For details, refer to the List of Error Codes.)

Description

This function cancels the association of the data with the registered smfID.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFUnbind(&smfCtx, smf, 0);

        return 0;
}
```

Notes

Since operations can no longer be performed for a canceled smfID, first stop playback for the smfID if it is being played before using this function.

sceSsSMFPlay

Start performance of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFPlay(
        SceUInt32 smfID,
        SceUInt32 playVelocity,
        SceUInt32 playPan,
        SceUInt32 playSendVelocity,
        SceUInt32 playSendPan,
        SceUInt32 playCount
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

smfID	smf ID for which performance is to be started
playVelocity	Performance velocity (0 - 127)
playPan	Performance panpot (0 - 64 - 127)
playSendVelocity	Performance send velocity (0 - 127)
playSendPan	Performance send panpot (0 - 64 - 127)
playCount	Number of performances (>= 1; set `SCE_SNDP_INFINITY` for continuous looping)

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function starts the performance of an SMF.

playVelocity is a relative value of volume for the entire performance.

playPan is the orientation for the entire performance.

playSendVelocity is the a relative value of the effect send volume for the entire performance.

playSendPan is the orientation for the effect for the entire performance.

playCount is the total number of performances. If `SCE_SNDP_INFINITY` is specified, playback will loop continuously.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;
        char *smf;

        result = sceSsSMFPlay(0, 127, 64, 127, 64, SCE_SNDP_INFINITY);

        return 0;
}
```

sceSsSMFPause

Pause performance of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFPause(
        SceUInt32 smfID
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

smfID	smf ID for which performance is to be paused

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function pauses the performance of an SMF that is being performed.

It turns off the suspend hold state of all sound-producing voices in the SMF to forcibly stop sound production.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFPause(0);

        return 0;
}
```

See Also

`sceSsSMFResume()`

sceSsSMFResume

Resume performance of paused SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFResume(
        SceUInt32 smfID
);
```

Calling Conditions

Can be called from an interrupt handler
Can be called from a thread (does not depend on interrupt-disabled or -enabled state)
Not multithread safe

Arguments

smfID	smf ID for which performance is to be resumed

Return Value

When the function completes normally, `SCE_OK` is returned.
If the return value is negative (< 0), it indicates an error.
(For details, see the List of Error Codes.)

Description

This function resumes the performance of the SMF from the point where it was paused by `sceSsSMFPause()`.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFResume(0);

        return 0;
}
```

sceSsSMFStop

Stop performance of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFStop(
         SceUInt32 smfID
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

smfID	smf ID for which performance is to be stopped

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function stops the performance of an SMF and returns its playback position to the beginning of the sequence data.

Example

```
#include <libsndp.h>

int main(void)
{
         int result;

         result = sceSsSMFStop(0);

         return 0;
}
```

sceSsSMFAddTempo

Add tempo to performance of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFAddTempo(
        SceUInt32 smfID,
        SceInt32 addTempo
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

smfID	smf ID for which tempo is to be added to performance
addTempo	Microseconds per quarter note to be added

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function adds tempo to the performance of an SMF. *addTempo* is the number of microseconds per quarter note to be added to the current tempo. If it is positive, playback slows down. If it is negative, playback speeds up.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFAddTempo(0, 10000);

        return 0;
}
```

sceSsSMFGetTempo

Get performance tempo of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFGetTempo(
        SceUInt32 smfID
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe

Arguments

smfID	smf ID for which tempo is to be obtained

Return Value

The number of microseconds per quarter note is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function gets the tempo of the performance. The return value is the number of microseconds per quarter note.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFGetTempo(0);

        return 0;
}
```

sceSsSMFSetPlayVelocity

Set performance velocity of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFSetPlayVelocity(
        SceUInt32 smfID,
        SceUInt32 playVelocity
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

smfID	smf ID for which performance velocity is to be set
playVelocity	Performance velocity (0 - 127)

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function sets the velocity for the entire SMF performance.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFSetPlayVelocity(0, 127);

        return 0;
}
```

sceSsSMFGetPlayVelocity

Get performance velocity of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFGetPlayVelocity(
        SceUInt32 smfID
);
```

Calling Conditions

Can be called from an interrupt handler
Can be called from a thread (does not depend on interrupt-disabled or -enabled state)
Multithread safe

Arguments

smfID	smf ID for which performance velocity is to be obtained

Return Value

The performance velocity is returned.
If the return value is negative (< 0), it indicates an error.
(For details, see the List of Error Codes.)

Description

This function gets the performance velocity.
Returns value set by `sceSsSMFSetPlayVelocity()`.

Example

```
int main(void)
{
        int result;

        result = sceSsSMFGetPlayVelocity(0);

        return 0;
}
```

sceSsSMFSetPlaySendVelocity

Set performance effect send velocity of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFSetPlaySendVelocity(
        SceUInt32 smfID,
        SceUInt32 playSendVelocity
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

smfID	smf ID for which performance velocity is to be set
playSendVelocity	Performance velocity (0 - 127)

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function sets the effect send velocity for the entire SMF performance.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFSetPlaySendVelocity(0, 127);

        return 0;
}
```

sceSsSMFGetPlaySendVelocity

Get performance effect send velocity of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFGetPlaySendVelocity(
        SceUInt32 smfID
);
```

Calling Conditions

Can be called from an interrupt handler
Can be called from a thread (does not depend on interrupt-disabled or -enabled state)
Multithread safe

Arguments

smfID	smf ID for which performance effect send velocity is to be obtained

Return Value

The performance effect send velocity is returned.
If the return value is negative (< 0), it indicates an error.
(For details, see the List of Error Codes.)

Description

This function gets the performance velocity.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFGetPlaySendVelocity(0);

        return 0;
}
```

sceSsSMFSetPlayPanpot

Set performance panpot of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFSetPlayPanpot(
        SceUInt32 smfID,
        SceUInt32 playPanpot
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

smfID	smf ID for which performance panpot is to be set
playPanpot	Performance panpot (0 - 64 - 127)

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function sets the orientation for the entire performance of the SMF.

0 corresponds to the leftmost position, 64 to the middle, and 127 to the rightmost position.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFSetPlayPanpot(0, 127);

        return 0;
}
```

sceSsSMFGetPlayPanpot

Get performance panpot of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFGetPlayPanpot(
        SceUInt32 smfID
);
```

Calling Conditions

Can be called from an interrupt handler
Can be called from a thread (does not depend on interrupt-disabled or -enabled state)
Multithread safe

Arguments

smfID	smf ID for which performance panpot is to be obtained

Return Value

The performance panpot is returned.
If the return value is negative (< 0), it indicates an error.
(For details, see the List of Error Codes.)

Description

This function gets the performance panpot.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFGetPlayPanpot(0);

        return 0;
}
```

sceSsSMFSetPlaySendPanpot

Set performance effect send panpot of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFSetPlaySendPanpot(
        SceUInt32 smfID,
        SceUInt32 playSendPanpot
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

smfID	smf ID for which the performance effect send panpot is to be set
playSendPanpot	Performance effect send panpot (0 - 64 - 127)

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function sets the effect send orientation for the entire performance of the SMF.

0 corresponds to the leftmost position, 64 to the middle, and 127 to the rightmost position.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFSetPlaySendPanpot(0, 127);

        return 0;
}
```

sceSsSMFGetPlaySendPanpot

Get performance effect send panpot of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFGetPlaySendPanpot(
        SceUInt32 smfID
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe

Arguments

smfID	smf ID for which performance effect send panpot is to be obtained

Return Value

The performance panpot is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function gets the performance effect send panpot.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFGetPlaySendPanpot(0);

        return 0;
}
```

sceSsSMFGetPlayStatus

Get performance status of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFGetPlayStatus(
        SceUInt32 smfID
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Multithread safe

Arguments

smfID	smf ID for which performance status is to be obtained

Return Value

The performance status is returned.

Macro	Description
SCE_SNDP_PLAYSTATUS_READY	Performance can be started
SCE_SNDP_PLAYSTATUS_PLAY	Performance is in progress
SCE_SNDP_PLAYSTATUS_STOP	Performance is stopped
SCE_SNDP_PLAYSTATUS_ERROR	Performance cannot take place

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function gets the performance status of an SMF.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFGetPlayStatus(0);

        return 0;
}
```

sceSsSMFSetChannelMute

Set channel mute of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFSetChannelMute(
        SceUInt32 smfID,
        SceUInt32 midiChannel
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

smfID	smf ID for which channel mute is to be set
midiChannel	MIDI channel to be muted (0 to 15 = MIDI channels 1 to 16)

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function mutes the performance of the MIDI channel specified by the SMF.

If sound is currently being produced, this function internally performs a key off, and then subsequently prohibits any key on from being performed.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFSetChannelMute(0, 0);

        return 0;
}
```

sceSsSMFResetChannelMute

Cancel channel mute of SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFResetChannelMute(
        SceUInt32 smfID
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

smfID	smf ID for which channel mute is to be canceled

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function cancels all channel muting.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFResetChannelMute(0);

        return 0;
}
```

sceSsSMFSetPlayChannel

Simultaneously set muting for all channels of an SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFSetPlayChannel(
        SceUInt32 smfID,
        SceUInt32 playChannelBit
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

smfID	smf ID for which channel muting is to be set
playChannelBit	Sets mute ON or OFF for all MIDI channels bits 0 to 15 = MIDI channels 1 to 16; bit value 0 = `Mute_ON`, 1 = `Mute_OFF`

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function sets the mute state of all MIDI channels of an SMF.

The bits of *`playChannelBit`* correspond to each of the MIDI channels, and the value of each bit indicates whether muting should be set ON or OFF for the respective channel.

If sound is being produced, the function internally performs a key off, then disables key on.

Example

```
#include <libsndp.h>

int main(void)
{
        int result;

        result = sceSsSMFSetPlayChannel(0, 0xFFFF); /* Cancel all muting */

        return 0;
}
```

sceSsSMFGetPlayChannel

Get the mute states for all channels of an SMF

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFGetPlayChannel(
        SceUInt32 smfID,
        SceUInt32 *playChannelBit
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

smfID	smf ID for which channel muting is to be obtained
playChannelBit	Pointer for storing mute states of all MIDI channels bits 0 to 15 = MIDI channels 1 to 16; bit value 0 = `Mute_ON`, 1 = `Mute_OFF`

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it indicates an error.

(For details, see the List of Error Codes.)

Description

This function gets the mute states of all MIDI channels of an SMF.

The bits of *playChannelBit* correspond to each of the MIDI channels, and the value of each bit indicates whether muting is turned ON or OFF for the respective channel.

Example

```
#include <libsndp.h>

/* Return mute states of each MIDI channel in MuteBit */
int func(unsigned int *MuteBit)
{
        int result;

        result = sceSsSMFGetPlayChannel(0, MuteBit);
        return result;
}
```

sceSsSMFGetKeyOnID

Get key on ID during an SMF performance

Definition

```
#include <libsndp.h>
SceInt32 sceSsSMFGetKeyOnID(
        SceUInt32 smfID,
        SceUInt32 midich,
        SceUInt32 *keyonID
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread (does not depend on interrupt-disabled or -enabled state)

Not multithread safe

Arguments

smfID	smfID that was obtained by `sceSsSMFBind()`
midich	MIDI channel
keyonID	ID value return value

Return Value

When the function completes normally, `SCE_OK` is returned.

If the return value is negative (< 0), it means that an error occurred.

(For details, see the List of Error Codes.)

Description

This function gets the key on ID value for an arbitrary *`midich`* during the performance of an SMF indicated by *`smfID`*.

Example

```
#include <libsndp.h>

/* Get key on ID of the SMF midich channel during a performance by smfID */
int GetSMFKeyOnVoiceBit(unsigned int smfID, unsigned int midich)
{
   int result;
   unsigned int keyonID;

   result = sceSsSMFGetKeyOnID (smfID, midich, &keyonID);
   return result;
}
```

Constants

List of Error Codes

List of error codes returned by libsndp

Definition

Macro	Value	Description
`SCE_SNDP_ERROR_NOTINIT`	0x80450001	Initialization has not been performed
`SCE_SNDP_ERROR_PHDID`	0x80450002	phd ID is invalid
`SCE_SNDP_ERROR_SMF`	0x80450003	Pointer to SMF data is invalid
`SCE_SNDP_ERROR_SMFCTX`	0x80450004	Pointer to SMF environment structure is invalid
`SCE_SNDP_ERROR_FORMAT`	0x80450005	SMF format is invalid
`SCE_SNDP_ERROR_SMFFULL`	0x80450006	Max. number of SMF registrations was exceeded
`SCE_SNDP_ERROR_SMFID`	0x80450007	smf ID is invalid
`SCE_SNDP_ERROR_PLAYCOUNT`	0x80450008	Specified performance count is invalid
`SCE_SNDP_ERROR_NOTREADY`	0x80450009	Status is such that performance cannot be started
`SCE_SNDP_ERROR_PARAM`	0x8045000a	Parameter is invalid
`SCE_SNDP_ERROR_NOTPLAY`	0x8045000b	Not being performed
`SCE_SNDP_ERROR_NOTPAUSE`	0x8045000c	Not paused
`SCE_SNDP_ERROR_DONTSTOP`	0x8045000d	Status is such that performance cannot be stopped
`SCE_SNDP_ERROR_ALREADY`	0x8045000e	Status has already been set
`SCE_SNDP_ERROR_SOUNDDATAFULL`	0x8045000f	Max. number of sound data registrations was exceeded
`SCE_SNDP_ERROR_CANTALLOCATEVOICE`	0x80450010	Voice cannot be allocated
`SCE_SNDP_ERROR_VOICENUM`	0x80450011	Voice number is invalid
`SCE_SNDP_ERROR_RESERVEDVOICE`	0x80450012	Voice is not being managed by libsndp
`SCE_SNDP_ERROR_PAUSEVOICE`	0x80450013	A paused voice is included