SAS Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

Document serial number: 000004892117

Constant Definitions

SCE_SAS_VOICE_MAX

Maximum number of voices

Definition

```
#include <sas.h>
#define SCE_SAS_VOICE_MAX           (32)
```

Description

This is the maximum number of voices.

See Also

`sceSasSetKeyOn()`

SCE_SAS_GRAIN_SAMPLES

Number of samples in one unit of granularity (default value)

Definition

```
#include <sas.h>
#define SCE_SAS_GRAIN_SAMPLES              (256)
```

Description

This is the number of samples that are generated whenever the periodic processing function is called. This value is different from the number of samples per unit of granularity specified with the `sceSasInitWithGrain()` function when SAS is initialized.

See Also

`sceSasCore()`, `sceSasInit()`, `sceSasInitWithGrain()`

SCE_SAS_VOLUME_MAX

Maximum volume

Definition

```
#include <sas.h>
#define SCE_SAS_VOLUME_MAX          (0x1000)
```

Description

This is the maximum volume that can be specified for each voice.

See Also

`sceSasSetVolume()`

SCE_SAS_LOOP_DISABLE

Disable loop flag

Definition

```
#include <sas.h>
#define SCE_SAS_LOOP_DISABLE (0)
```

Description

This is specified to disable the loop flag that is contained in the ADPCM (.vag) data.

If `SCE_SAS_LOOP_DISABLE` is specified, one-shot play is performed unconditionally.

See Also

`sceSasSetVoice()`

SCE_SAS_LOOP_ENABLE

Enable loop flag

Definition

```
#include <sas.h>
#define SCE_SAS_LOOP_ENABLE  (1)
```

Description

This is specified to enable the loop flag that is contained in the ADPCM (.vag) data.

If `SCE_SAS_LOOP_ENABLE` is specified, infinite loop play is performed for ADPCM (.vag) data that includes loop information. If the ADPCM (.vag) data does not include loop information, one-shot play is performed.

See Also

`sceSasSetVoice()`

SCE_SAS_PITCH_MIN

Pitch lower bound

Definition

```
#include <sas.h>
#define SCE_SAS_PITCH_MIN          (1)
```

Description

This is the lower bound of the pitch that can be specified.

See Also

`sceSasSetPitch()`

SCE_SAS_PITCH_BASE

Pitch base value

Definition

```
#include <sas.h>
#define SCE_SAS_PITCH_BASE          (0x1000)
```

Description

This is the pitch value that is specified when you don't want to change the pitch.

See Also

`sceSasSetPitch()`

SCE_SAS_PITCH_MAX

Pitch upper bound

Definition

```
#include <sas.h>
#define SCE_SAS_PITCH_MAX           (0x4000)
```

Description

This is the upper bound of the pitch that can be specified.

See Also

`sceSasSetPitch()`

SCE_SAS_NOISE_CLOCK_MAX

Noise clock upper bound

Definition

```
#include <sas.h>
#define SCE_SAS_NOISE_CLOCK_MAX     (0x3f)
```

Description

This is the upper bound of the noise clock value that can be specified.

See Also

`sceSasSetNoise()`

SCE_SAS_ENVELOPE_HEIGHT_MAX

Maximum envelope wave height

Definition

```
#include <sas.h>
#define SCE_SAS_ENVELOPE_HEIGHT_MAX (0x40000000)
```

Description

This is the maximum envelope wave height.

See Also

`sceSasSetSL()`

SCE_SAS_ENVELOPE_RATE_MAX

Maximum envelope rate value

Definition

```
#include <sas.h>
#define SCE_SAS_ENVELOPE_RATE_MAX          (0x7fffffff)
```

Description

This is the maximum envelope rate value.

See Also

`sceSasSetADSR()`

SCE_SAS_ADSR_MODE_LINEAR_INC

ADSR linear increase

Definition

```
#include <sas.h>
#define SCE_SAS_ADSR_MODE_LINEAR_INC       (0)
```

Description

This is a linear increase curve in the envelope ADSR curve specification.

See Also

`sceSasSetADSRmode()`

SCE_SAS_ADSR_MODE_LINEAR_DEC

ADSR linear decrease

Definition

```
#include <sas.h>
#define SCE_SAS_ADSR_MODE_LINEAR_DEC       (1)
```

Description

This is a linear decrease curve in the envelope ADSR curve specification.

See Also

`sceSasSetADSRmode()`

SCE_SAS_ADSR_MODE_BENT_LINEAR

ADSR bent linear increase

Definition

```
#include <sas.h>
#define SCE_SAS_ADSR_MODE_BENT_LINEAR      (2)
```

Description

This is a bent linear increase curve in the envelope ADSR curve specification.

See Also

`sceSasSetADSRmode()`

SCE_SAS_ADSR_MODE_REVEXPONENT

ADSR exponential decrease

Definition

```
#include <sas.h>
#define SCE_SAS_ADSR_MODE_REVEXPONENT      (3)
```

Description

This is an exponential decrease curve in the envelope ADSR curve specification.

See Also

`sceSasSetADSRmode()`

SCE_SAS_ADSR_MODE_EXPONENT

ADSR exponential increase

Definition

```
#include <sas.h>
#define SCE_SAS_ADSR_MODE_EXPONENT  (4)
```

Description

This is an exponential increase curve in the envelope ADSR curve specification.

See Also

`sceSasSetADSRmode()`

SCE_SAS_ADSR_MODE_DIRECT

ADSR immediate specification

Definition

```
#include <sas.h>
#define SCE_SAS_ADSR_MODE_DIRECT    (5)
```

Description

This is an immediate specification mode in the envelope ADSR specification.

See Also

`sceSasSetADSRmode()`

SCE_SAS_ATTACK_VALID

Make ADSR parameter change Attack part valid

Definition

```
#include <sas.h>
#define SCE_SAS_ATTACK_VALID (1)
```

Description

This is a flag indicating that a parameter change in the attack state is valid in functions for collectively setting all 4 ADSR state parameters of the envelope.

See Also

`sceSasSetADSR(),sceSasSetADSRmode()`

SCE_SAS_DECAY_VALID

Make ADSR parameter change Decay part valid

Definition

```
#include <sas.h>
#define SCE_SAS_DECAY_VALID (2)
```

Description

This is a flag indicating that a parameter change in the decay state is valid in functions for collectively setting all 4 ADSR state parameters of the envelope.

See Also

`sceSasSetADSR(),sceSasSetADSRmode()`

SCE_SAS_SUSTAIN_VALID

Make ADSR parameter change Sustain part valid

Definition

```
#include <sas.h>
#define SCE_SAS_SUSTAIN_VALID       (4)
```

Description

This is a flag indicating that a parameter change in the sustain state is valid in functions for collectively setting all 4 ADSR state parameters of the envelope.

See Also

`sceSasSetADSR(),sceSasSetADSRmode()`

SCE_SAS_RELEASE_VALID

Make ADSR parameter change Release part valid

Definition

```
#include <sas.h>
#define SCE_SAS_RELEASE_VALID       (8)
```

Description

This is a flag indicating that a parameter change in the release state is valid in functions for collectively setting all 4 ADSR state parameters of the envelope.

See Also

`sceSasSetADSR(),sceSasSetADSRmode()`

SCE_SAS_OUTPUTMODE_STEREO, SCE_SAS_OUTPUTMODE_MULTI

Output mode

Definition

```
#include <sas.h>
#define SCE_SAS_OUTPUTMODE_STEREO   (0)
#define SCE_SAS_OUTPUTMODE_MULTI    (1)
```

Description

This is the output mode identifier. `SCE_SAS_OUTPUTMODE_STEREO` corresponds to stereo mode (2-channel output), and `SCE_SAS_OUTPUTMODE_MULTI` corresponds to multichannel mode (4-channel output).

See Also

`sceSasSetOutputmode(), sceSasGetOutputmode()`

SCE_SAS_FX_VOLUME_MAX

Maximum effect volume

Definition

```
#include <sas.h>
#define SCE_SAS_FX_VOLUME_MAX              (0x1000)
```

Description

This is the maximum effect output volume value.

See Also

`sceSasSetEffectVolume()`

SCE_SAS_FX_TYPE_OFF

No effects

Definition

```
#include <sas.h>
#define SCE_SAS_FX_TYPE_OFF  (-1)
```

Description

This is a constant that indicates "no effects (through)" when specifying the effect format.

See Also

`sceSasSetEffectType()`

SCE_SAS_FX_TYPE_ROOM

Effect type Room

Definition

```
#include <sas.h>
#define SCE_SAS_FX_TYPE_ROOM (0)
```

Description

This is a constant that indicates "Room" when specifying the effect format.

See Also

`sceSasSetEffectType()`

SCE_SAS_FX_TYPE_STUDIOA

Effect type Studio-A

Definition

```
#include <sas.h>
#define SCE_SAS_FX_TYPE_STUDIOA     (1)
```

Description

This is a constant that indicates "Studio-A" when specifying the effect format.

See Also

`sceSasSetEffectType()`

SCE_SAS_FX_TYPE_STUDIOB

Effect type Studio-B

Definition

```
#include <sas.h>
#define SCE_SAS_FX_TYPE_STUDIOB     (2)
```

Description

This is a constant that indicates "Studio-B" when specifying the effect format.

See Also

`sceSasSetEffectType()`

SCE_SAS_FX_TYPE_STUDIOC

Effect type Studio-C

Definition

```
#include <sas.h>
#define SCE_SAS_FX_TYPE_STUDIOC     (3)
```

Description

This is a constant that indicates "Studio-C" when specifying the effect format.

See Also

`sceSasSetEffectType()`

SCE_SAS_FX_TYPE_HALL

Effect type Hall

Definition

```
#include <sas.h>
#define SCE_SAS_FX_TYPE_HALL (4)
```

Description

This is a constant that indicates "Hall" when specifying the effect format.

See Also

`sceSasSetEffectType()`

SCE_SAS_FX_TYPE_SPACE

Effect type Space

Definition

```
#include <sas.h>
#define SCE_SAS_FX_TYPE_SPACE       (5)
```

Description

This is a constant that indicates "Space" when specifying the effect format.

See Also

`sceSasSetEffectType()`

SCE_SAS_FX_TYPE_ECHO

Effect type Echo

Definition

```
#include <sas.h>
#define SCE_SAS_FX_TYPE_ECHO (6)
```

Description

This is a constant that indicates "Echo" when specifying the effect format.

See Also

`sceSasSetEffectType()`

SCE_SAS_FX_TYPE_DELAY

Effect type Delay

Definition

```
#include <sas.h>
#define SCE_SAS_FX_TYPE_DELAY       (7)
```

Description

This is a constant that indicates "Delay" when specifying the effect format.

See Also

`sceSasSetEffectType()`

SCE_SAS_FX_TYPE_PIPE

Effect type Pipe

Definition

```
#include <sas.h>
#define SCE_SAS_FX_TYPE_PIPE        (8)
```

Description

This is a constant that indicates "Pipe" when specifying the effect format.

See Also

`sceSasSetEffectType()`

Initialization/Termination Functions

sceSasGetNeededMemorySize

Get memory size (in bytes) required for initialization

Definition

```
#include <sas.h>
SceSasResult sceSasGetNeededMemorySize(
        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 "")
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 Error Codes.)

Description

This function obtains the memory size required for SAS initialization.

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

Examples

```
SceSize bufferSize;
void *buffer;
SceSasResult result;

/* Obtain the memory size required for SAS initialization */
result = sceSasGetNeededMemorySize("", &bufferSize);
if (result < 0) {
        printf("Error: sceSasGetNeededMemorySize() %X\n", result);
        exit(EXIT_FAILURE);
}

/* Allocate the memory required for SAS initialization */
buffer = malloc(bufferSize);
if (buffer == NULL) {
        printf("Error: Failed to allocate memory %d\n", bufferSize);
        exit(EXIT_FAILURE);
}

/* Initialize SAS */
result = sceSasInit("", buffer, bufferSize);
if (result < 0) {
        printf("Error: sceSasInit() %X\n", result);
        exit(EXIT_FAILURE);
}
```

sceSasInit

Initialization processing

Definition

```
#include <sas.h>
SceSasResult sceSasInit(
        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

config	For future expansion (currently specify NULL)
buffer	Pointer to initialization buffer
bufferSize	Initialization buffer size

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 Error Codes.)

Description

This function initializes the SAS state.

To use SAS, this function or `sceSasInitWithGrain()` must be called.

Be sure to use `sceSasGetNeededMemorySize()` to obtain the memory size required for SAS initialization.

When SAS is initialized by using this function, the number of PCM samples (granularity) that are generated by a single call of the `sceSasCore()` and `sceSasCoreWithMix()` functions is `SCE_SAS_GRAIN_SAMPLES(=256)`. Also, the output mode is initialized as stereo mode.

Examples

```
SceSize bufferSize;
void *buffer;
SceSasResult result;

/* Obtain the memory size required for SAS initialization */
result = sceSasGetNeededMemorySize("", &bufferSize);
if (result < 0) {
        printf("Error: sceSasGetNeededMemorySize() %X\n", result);
        exit(EXIT_FAILURE);
}

/* Allocate the memory required for SAS initialization */
buffer = malloc(bufferSize);
if (buffer == NULL) {
        printf("Error: Failed to allocate memory %d\n", bufferSize);
        exit(EXIT_FAILURE);
}

/* Initialize SAS */
result = sceSasInit("", buffer, bufferSize);
if (result < 0) {
        printf("Error: sceSasInit() %X\n", result);
        exit(EXIT_FAILURE);
}
```

sceSasInitWithGrain

Initialization processing with granularity specification

Definition

```
#include <sas.h>
SceSasResult sceSasInitWithGrain(
        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

config	For future expansion (currently specify NULL)
grain	Number of samples per channel that are generated by one unit of granularity (values can range from 64 to 2048 in multiples of 32)
buffer	Pointer to initialization buffer
bufferSize	Initialization buffer size

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 Error Codes.)

Description

This function initializes the SAS state.

To use SAS, this function or `sceSasInit()` must be called.

Be sure to use `sceSasGetNeededMemorySize()` to obtain the memory size required for SAS initialization.

Using this function to initialize SAS enables you to set the number of PCM samples (granularity) that are generated by a single call of the `sceSasCore()` and `sceSasCoreWithMix()` functions to a value other than `SCE_SAS_GRAIN_SAMPLES(=256)`. Also, the output mode is initialized as stereo mode.

Examples

```
SceSize bufferSize;
void *buffer;
SceSasResult result;

/* Obtain the memory size required for SAS initialization */
result = sceSasGetNeededMemorySize("", &bufferSize);
if (result < 0) {
        printf("Error: sceSasGetNeededMemorySize() %X\n", result);
        exit(EXIT_FAILURE);
}

/* Allocate the memory required for SAS initialization */
buffer = malloc(bufferSize);
if (buffer == NULL) {
        printf("Error: Failed to allocate memory %d\n", bufferSize);
        exit(EXIT_FAILURE);
}

/* Initialize SAS with granularity as 1024 */
result = sceSasInitWithGrain("", 1024, buffer, bufferSize);
if (result < 0) {
        printf("Error: sceSasInitWithGrain() %X\n", result);
        exit(EXIT_FAILURE);
}
```

sceSasExit

Termination processing

Definition

```
#include <sas.h>
SceSasResult sceSasExit(
        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 Values

When the function completes normally, then SCE_OK is returned.
When an error occurs, a negative value (< 0) is returned.
(For details, see Error Codes.)

Description

This function performs termination processing.
When an application that uses SAS ends, be sure to execute sceSasExit().

Examples

```
SceSize bufferSize;
void *buffer;
SceSasResult result;

/* Terminate SAS */
result = sceSasExit(&buffer, &bufferSize);
if (result < 0) {
        printf("Error: sceSasExit() %X\n", result);
}

/* Free SAS work area */
free(buffer);
```

Sound Functions

sceSasCore

SAS periodic processing

Definition

```
#include <sas.h>
SceSasResult sceSasCore(
        SceInt16 *out
);
```

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.

Arguments

out Pointer to waveform output buffer

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 Error Codes.)

Description

When this function is called, a waveform for one unit of granularity is output. The number of samples in one unit of granularity is SCE_SAS_GRAIN_SAMPLES when SAS is initialized with the sceSasInit() function. When SAS is initialized with the sceSasInitWithGrain() function, the number of samples in one unit of granularity is the value that was specified with the function.

Since this function is multithread safe, another function such as sceSasSetKeyOn(), for example, can be called while this function is being executed, however the sceSasCore() and sceSasCoreWithMix() functions themselves cannot be called simultaneously from multiple threads (SCE_SAS_ERROR_BUSY will be returned).

Examples

```
static SceInt16 aPcmBuffer[2][256]; /* Set up a double buffer */
static int bufferId = 0;
SceSasResult result;

while (1) {
        /* Create a waveform for one unit of granularity */
        result = sceSasCore(aPcmBuffer[bufferId]);
        if (result < 0) {
              printf("Error: sceSasCore() %X\n", result);
        }

        /* Switch double buffer ID */
        bufferId ^= 1;

        /* Output a waveform for one unit of granularity */
              :
}
```

sceSasCoreWithMix

SAS periodic processing with external PCM mixing function

Definition

```
#include <sas.h>
SceSasResult sceSasCoreWithMix(
        SceInt16 *inOut,
        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.

Arguments

inOut	Pointer to waveform output buffer
lvol	External PCM input left channel volume
rvol	External PCM input right channel volume

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 Error Codes.)

Description

When this function is called, a waveform for one unit of granularity is synthesized, overwritten by the result obtained by mixing it with PCM data that is in a specified area, and output. The number of samples in one unit of granularity is SCE_SAS_GRAIN_SAMPLES when SAS is initialized with the sceSasInit() function. When SAS is initialized with the sceSasInitWithGrain() function, the number of samples in one unit of granularity is the value that was specified with the function.

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

$0 \leqq lvol, rvol \leqq$ SCE_SAS_VOLUME_MAX

Since this function is multithread safe, another function such as sceSasSetKeyOn(), for example, can be called while this function is being executed, however the sceSasCore() and sceSasCoreWithMix() functions themselves cannot be called simultaneously from multiple threads (SCE_SAS_ERROR_BUSY will be returned).

Examples

```
static SceInt16 aPcmBuffer[2][256]; /* Set a double buffer */
static int bufferId = 0;
SceSasResult result;

while (1) {
        /* Waveform output processing for one unit of granularity */
        result = sceSasCoreWithMix(
              aPcmBuffer[bufferId],
              SCE_SAS_VOLUME_MAX,
              SCE_SAS_VOLUME_MAX);
        if (result < 0) {
              printf("Error: sceSasCoreWithMix() %X\n", result);
        }

        /* Switch double buffer ID */
        bufferId ^= 1;

        /* Output a waveform for one unit of granularity */
              :
}
```

sceSasSetKeyOn

Key on

Definition

```
#include <sas.h>
SceSasResult sceSasSetKeyOn(
        SceInt32 iVoiceNum
);
```

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.

Arguments

iVoiceNum Voice number (0 - 31)

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 Error Codes.)

Description

This function performs key on for a voice.

Note
If key-on is performed for a voice that is paused, an error occurs.

sceSasSetKeyOff

Key off

Definition

```
#include <sas.h>
SceSasResult sceSasSetKeyOff(
        SceInt32 iVoiceNum
);
```

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.

Arguments

iVoiceNum Voice number (0 - 31)

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 Error Codes.)

Description

This function performs key off for a voice.

Note
If key-off is performed for a voice that is paused, an error occurs.

sceSasSetPause

Change pause state

Definition

```
#include <sas.h>
SceSasResult sceSasSetPause(
        SceInt32 iVoiceNum,
        SceUInt32 pauseFlag
);
```

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.

Arguments

iVoiceNum	Voice number for which pause state is to be changed
pauseFlag	Pause state (1: Enable pause; 0: Disable pause)

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 Error Codes.)

Description

This function sets or cancels the pause state of a voice with the specified number.

sceSasGetPauseState

Get pause state of each voice

Definition

```
#include <sas.h>
SceSasResult sceSasGetPauseState(
        SceInt32 iVoiceNum
);
```

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.

Arguments

iVoiceNum Voice number from which pause state is to be obtained

Return Values

Pause state
When an error occurs, a negative value (< 0) is returned.
(For details, see Error Codes.)

Description

This function obtains the pause state of a voice.
1 is returned for a voice that is paused, and 0 is for a voice that is not paused.

Examples

```
SceSasResult result;

/* Check pause state */
result = sceSasGetPauseState(0);
if (0 <= result) {
        printf("Pause State = %s\n", (result)? "Paused" : "None");
} else {
        printf("Error: sceSasGetPauseState() %X\n", result);
}
```

sceSasSetVolume

Set volume

Definition

```
#include <sas.h>
SceSasResult sceSasSetVolume(
        SceInt32 iVoiceNum,
        SceInt32 l,
        SceInt32 r,
        SceInt32 wl,
        SceInt32 wr
);
```

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.

Arguments

iVoiceNum	Voice number (0 - 31)
l	Left volume
r	Right volume
wl	Left effect volume
wr	Right effect volume

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 Error Codes.)

Description

This function sets the volume of a voice.

The range that can be set as a volume value is given by the following expression.

$$-\text{SCE\_SAS\_VOLUME\_MAX} \leqq l, r, wl, wr \leqq \text{SCE\_SAS\_VOLUME\_MAX}$$

The phase is inverted for a negative number.

In multichannel mode, the volume values set by this function determine the volume assignments for the four channels of output from each voice.

sceSasSetPitch

Set pitch

Definition

```
#include <sas.h>
SceSasResult sceSasSetPitch(
        SceInt32 iVoiceNum,
        SceInt32 pitch
);
```

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.

Arguments

iVoiceNum	Voice number (0 - 31)
pitch	Pitch (SCE_SAS_PITCH_MIN - SCE_SAS_PITCH_MAX)

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 Error Codes.)

Description

This function sets the pitch of a voice.

sceSasSetVoice

Set voice (VAG format)

Definition

```
#include <sas.h>
SceSasResult sceSasSetVoice(
        SceInt32 iVoiceNum,
        const void *vagBuf,
        SceSize size,
        SceUInt32 loopFlag
);
```

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.

Arguments

iVoiceNum	Voice number (0 - 31)
vagBuf	Pointer to waveform data buffer
size	Waveform data size
loopFlag	Loop flag (SCE_SAS_LOOP_ENABLE or SCE_SAS_LOOP_DISABLE)

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 Error Codes.)

Description

This function sets information related to waveform data (VAG format) for which sound is to be produced by a voice.

sceSasSetVoicePCM

Set voice (PCM format)

Definition

```
#include <sas.h>
SceSasResult sceSasSetVoicePCM(
        SceInt32 iVoiceNum,
        const void *pcmBuf,
        SceSize size,
        SceInt32 loopSize
);
```

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.

Arguments

iVoiceNum	Voice number (0 - 31)
pcmBuf	Pointer to waveform data buffer
size	Waveform data size (number of samples)
loopSize	Number of samples from the beginning to the loop starting point. To disable looping, specify a negative number.

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 Error Codes.)

Description

This function sets information related to PCM data for which sound is to be produced by a voice.

The function expects the PCM sound source data to be 16-bit, monaural, linear PCM. To loop part of the PCM sound source data, specify the number of samples from the beginning of the data to the loop starting point for *loopSize*. After the PCM sound source is played to the end, playback will return to the loop starting point and continue.

sceSasSetNoise

Set noise

Definition

```
#include <sas.h>
SceSasResult sceSasSetNoise(
        SceInt32 iVoiceNum,
        SceUInt32 clock
);
```

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.

Arguments

iVoiceNum	Voice number (0 - 31)
clock	Noise clock (0x00 - 0x3f)

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 Error Codes.)

Description

This function sets the noise clock and performs processing so that sound can be produced by noise.

sceSasSetADSR

Set envelope (ADSR)

Definition

```
#include <sas.h>
SceSasResult sceSasSetADSR(
        SceInt32 iVoiceNum,
        SceUInt32 flag,
        SceUInt32 a,
        SceUInt32 d,
        SceUInt32 s,
        SceUInt32 r
);
```

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.

Arguments

iVoiceNum	Voice number (0 - 31)
flag	Setting flag
a	Attack rate (0 - SCE_SAS_ENVELOPE_RATE_MAX)
d	Decay rate (0 - SCE_SAS_ENVELOPE_RATE_MAX)
s	Sustain rate (0 - SCE_SAS_ENVELOPE_RATE_MAX)
r	Release rate (0 - SCE_SAS_ENVELOPE_RATE_MAX)

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 Error Codes.)

Description

This function sets the time change of an envelope of a voice. Specify a logical OR of SCE_SAS_ATTACK_VALID, SCE_SAS_DECAY_VALID, SCE_SAS_SUSTAIN_VALID, and SCE_SAS_RELEASE_VALID for *flag*. Only an envelope of the states specified by *flag* is changed.

If an out-of-range value is set for *a*, *d*, *s*, or *r*, an error occurs.

For details about the values for the attack rate, decay rate, sustain rate, and release rate, refer to the "SAS Overview" document.

sceSasSetADSRmode

Set envelope (ADSR) curve type

Definition

```
#include <sas.h>
SceSasResult sceSasSetADSRmode(
        SceInt32 iVoiceNum,
        SceUInt32 flag,
        SceInt32 a,
        SceInt32 d,
        SceInt32 s,
        SceInt32 r
);
```

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.

Arguments

iVoiceNum	Voice number (0 - 31)
flag	Setting flag
a	Attack curve type
d	Decay curve type
s	Sustain curve type
r	Release curve type

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 Error Codes.)

Description

This function sets the envelope curve type of a voice. Specify a logical OR of SCE_SAS_ATTACK_VALID, SCE_SAS_DECAY_VALID, SCE_SAS_SUSTAIN_VALID, and SCE_SAS_RELEASE_VALID for *flag*. Only the curve types of the states specified by *flag* are changed.

The following values can be set for the various curve types.

For details about curves, refer to the "SAS Overview" document.

Curve Type Value	Description
SCE_SAS_ADSR_MODE_LINEAR_INC	Linear increase (+lin)
SCE_SAS_ADSR_MODE_LINEAR_DEC	Linear decrease (-lin)
SCE_SAS_ADSR_MODE_BENT_LINEAR	Broken line increase (+bent lin)
SCE_SAS_ADSR_MODE_REVEXPONENT	Exponential function decrease (-exp)
SCE_SAS_ADSR_MODE_EXPONENT	Exponential function increase (+exp)
SCE_SAS_ADSR_MODE_DIRECT	Direct value specification (direct)

sceSasSetSL

Set envelope sustain level

Definition

```
#include <sas.h>
SceSasResult sceSasSetSL(
        SceInt32 iVoiceNum,
        SceInt32 sl
);
```

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.

Arguments

iVoiceNum	Voice number (0 - 31)
sl	Sustain level (0 - SCE_SAS_ENVELOPE_HEIGHT_MAX)

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 Error Codes.)

Description

This function sets the sustain level in the envelope of a voice.
If an out-of-range value is set, an error occurs.

sceSasSetSimpleADSR

Simple setting of envelope (ADSR)

Definition

```
#include <sas.h>
SceSasResult sceSasSetSimpleADSR(
        SceInt32 iVoiceNum,
        SceUInt16 adsr1,
        SceUInt16 adsr2
);
```

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.

Arguments

iVoiceNum	Voice number (0 - 31)
adsr1	Envelope value 1
adsr2	Envelope value 2

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 Error Codes.)

Description

This function sets the envelope of a voice.
The *adsr1* and *adsr2* values are as described in the "SAS Overview" document.

sceSasGetEndState

Get sound generation end state

Definition

```
#include<sas.h>
SceSasResult sceSasGetEndState(
        SceInt32 iVoiceNum
);
```

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.

Arguments

iVoiceNum Voice number (0 - 31)

Return Values

The voice states are returned.
When an error occurs, a negative value (< 0) is returned.
(For details, see Error Codes.)

Description

This function gets the sound generation end state of a voice.
1 is returned for a voice for which sound generation ended. If sound is being generated, 0 is returned.

Examples

```
SceSasResult result;

/* Check sound generation state */
result = sceSasGetEndState(0);
if (0 <= result) {
        printf("End State = %s\n", (result)? "End" : "Playing");
} else {
        printf("Error: sceSasGetEndState() %X\n", result);
}
```

sceSasGetEnvelope

Get envelope value

Definition

```
#include <sas.h>
SceSasResult sceSasGetEnvelope(
        SceInt32 iVoiceNum
);
```

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.

Arguments

iVoiceNum Voice number (0 - 31)

Return Values

The envelope value is returned.
When an error occurs, a negative value (< 0) is returned.
(For details, see Error Codes.)

Description

This function gets the envelope value of a voice.

Examples

```
SceSasResult result;

/* Check envelope */
result = sceSasGetEnvelope(0);
if (0 <= result) {
        printf("Envelope height = %d\n", result);
} else {
        printf("Error: sceSasGetEnvelope() %X\n", result);
}
```

sceSasSetGrain

Set granularity

Definition

```
#include <sas.h>
SceSasResult sceSasSetGrain(
        SceUInt32 grain
);
```

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.

Arguments

grain Number of samples per channel that are generated during one unit of granularity (multiple of 32 from 64 to 2048)

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 Error Codes.)

Description

This function changes the number of PCM samples (the granularity) that are generated by a single call of the `sceSasCore()` and `sceSasCoreWithMix()` functions. Note that the output buffer size that is required by the `sceSasCore()` and `sceSasCoreWithMix()` functions changes whenever the granularity is changed. If the buffer size is insufficient, there is a risk that output data will be written in an illegal area.

Examples

```
SceSasResult result;

/* Set the granularity to 1024 */
result = sceSasSetGrain(1024);
if (result < 0) {
        printf("Error: sceSasSetGrain() %X\n", result);
}
```

sceSasGetGrain

Get granularity

Definition

```
#include <sas.h>
SceSasResult sceSasGetGrain(
        void
);
```

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.

Arguments

None

Return Values

The granularity is returned. This value is a multiple of 32 from 64 to 2048.
When an error occurs, a negative value (< 0) is returned.
(For details, see Error Codes.)

Description

This function gets the number of PCM samples (the granularity) that are generated by a single call of the `sceSasCore()` and `sceSasCoreWithMix()` functions.

Examples

```
SceSasResult result;

/* Output current granularity to standard output */
result = sceSasGetGrain(&grain);
if (0 <= result) {
        printf("Grain = %d\n", result);
} else {
        printf("Error: sceSasGetGrain() %X\n", result);
}
```

sceSasSetOutputmode

Set output mode

Definition

```
#include <sas.h>
SceSasResult sceSasSetOutputmode(
        SceUInt32 outputmode
);
```

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.

Arguments

outputmode	Output mode (SCE_SAS_OUTPUTMODE_STEREO or SCE_SAS_OUTPUTMODE_MULTI)

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 Error Codes.)

Description

This function changes the output mode. The argument can be either SCE_SAS_OUTPUTMODE_STEREO, which corresponds to stereo mode (2-channel output with effects), or SCE_SAS_OUTPUTMODE_MULTI, which corresponds to multichannel mode (4-channel output with no effects). For details about each output mode, refer to the "SAS Overview" document.

In multichannel mode, a 4-channel monaural voice is output before adding effects. As a result, no effects processing is performed. If an effects setting function is called at this time, the effects setting function will return an error.

Since the number of output channels for multichannel mode is double the number for stereo mode, the output buffer size that is required for sceSasCore() and sceSasCoreWithMix() is also doubled. Note that if the buffer size is insufficient, there is a risk that output data will be written in an illegal area.

Use sceSasSetVolume() to assign volume to each voice for the four channels in multichannel mode.

In multichannel mode, the ordering of output PCM data is different from that of stereo mode, and the output PCM data that is obtained by sceSasCore() and sceSasCoreWithMix() cannot be passed directly to a sound output library such as an audio output function. Be sure to perform independent effects, mixing, and interleave processing on multichannel mode output PCM data before passing it to a library such as an audio output function.

Examples

```
SceSasResult result;

/* Set the output mode to multichannel mode */
result = sceSasSetOutputmode(SCE_SAS_OUTPUTMODE_MULTI);
if (result < 0) {
        printf("Error: sceSasSetOutputmode() %X\n", result);
}
```

sceSasGetOutputmode

Get output mode

Definition

```
#include <sas.h>
SceSasResult sceSasGetOutputmode(
        void
);
```

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.

Arguments

None

Return Values

The output mode identifier is returned.
When an error occurs, a negative value (< 0) is returned.
(For details, see Error Codes.)

Description

This function gets the identifier of the output mode that is currently set.

Examples

```
SceSasResult result;

/* Display the current output mode */
result = sceSasGetOutputmode();
if (result == SCE_SAS_OUTPUTMODE_STEREO) {
        printf("Stereo mode\n");
} else if (result == SCE_SAS_OUTPUTMODE_MULTI) {
        printf("Multi-channel mode\n");
} else if (0 <= result) {
        printf("Unknown output mode %d\n", result);
} else {
        printf("Error: sceSasGetOutputmode() %X\n", result);
}
```

Effect Setting Functions

sceSasSetEffectType

Set effect type

Definition

```
#include <sas.h>
SceSasResult sceSasSetEffectType(
        SceInt32 type
);
```

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.

Arguments

type Effect type

Return Values

When the function completes normally, then SCE_OK is returned.
If the *type* value is invalid, SCE_SAS_ERROR_FX_TYPE is returned.

Description

This function sets the effect type. The *type* is specified using the constant SCE_SAS_FX_XXX.
If the effect type is changed, the effect is reset (previous effects such as reverb are eliminated).
In multichannel mode, this function returns an error without setting the parameter.

The following values are set for the effect type.

Effect type value	Description
SCE_SAS_FX_TYPE_OFF	OFF
SCE_SAS_FX_TYPE_ROOM	Room Reverb
SCE_SAS_FX_TYPE_STUDIOA	Studio Reverb A
SCE_SAS_FX_TYPE_STUDIOB	Studio Reverb B
SCE_SAS_FX_TYPE_STUDIOC	Studio Reverb C
SCE_SAS_FX_TYPE_HALL	Hall Reverb
SCE_SAS_FX_TYPE_SPACE	Space Echo
SCE_SAS_FX_TYPE_ECHO	Echo
SCE_SAS_FX_TYPE_DELAY	Delay
SCE_SAS_FX_TYPE_PIPE	Pipe Echo

Examples

```
SceSasResult result;

/* Set the effect type to HALL */
result = sceSasSetEffectType(SCE_SAS_FX_TYPE_HALL);
if (result < 0) {
        printf("Error: sceSasSetEffectType() %X\n", result);
}
```

sceSasSetEffectParam

Effect parameters

Definition

```
#include<sas.h>
SceSasResult sceSasSetEffectParam(
        SceUInt32 dt,
        SceUInt32 fb
);
```

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.

Arguments

dt	Delay time (0 - 127)
fb	Feedback level (0 - 127)

Return Values

When the function completes normally, then SCE_OK is returned.

If the delay value is invalid (out of range), then SCE_SAS_ERROR_FX_DELAY is returned.

If the feedback value is invalid (out of range), then SCE_SAS_ERROR_FX_FEEDBACK is returned.

Description

This setting is valid only when SCE_SAS_FX_TYPE_ECHO or SCE_SAS_FX_TYPE_DELAY is set for the effect type.

If another type was specified, the value set here is ignored.

The value of argument *dt* is 0 to 127, where a larger value corresponds to a longer delay time.

The value of argument *fb* is 0 to 127, where a larger value corresponds to a greater amount of feedback.

In multichannel mode, this function returns an error without setting the parameter.

Note
If the delay time is changed while sound is being produced, noise may occur.

Examples

```
SceSasResult result;

/* Set the delay time and feedback of an effect to 64 and 80 respectively */
result = sceSasSetEffectParam(64, 80);
if (result < 0) {
        printf("Error: sceSasSetEffectParam() %X\n", result);
}
```

sceSasSetEffectVolume

Effect volume

Definition

```
#include <sas.h>
SceSasResult sceSasSetEffectVolume(
        SceInt32 val_l,
        SceInt32 val_r
);
```

Calling Conditions

Cannot be called from an interrupt handler.
Can be called from a thread (must be called in an interrupt-enabled state).
Not multithread safe.

Arguments

val_l	Left channel effect volume (0 to SCE_SAS_FX_VOLUME_MAX)
val_r	Right channel effect volume (0 to SCE_SAS_FX_VOLUME_MAX)

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 Error Codes.)

Description

This function sets the volume of the effect sound after effects processing.
The values of the left and right channels can be set using *val_l* and *val_r* respectively.
In multichannel mode, this function returns an error without setting the parameter.

Examples

```
SceSasResult result;

/* Set the left effect volume to 50% and right to 100% */
result = sceSasSetEffectVolume(2048, 4096);
if (result < 0) {
        printf("Error: sceSasSetEffectVolume() %X\n", result);
}
```

sceSasSetEffect

Effect switch

Definition

```
#include<sas.h>
SceSasResult sceSasSetEffect(
        SceInt32 dry_sw,
        SceInt32 wet_sw
);
```

Calling Conditions

Cannot be called from an interrupt handler.

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

Not multithread safe.

Arguments

dry_sw	Dry-side sound ON/OFF state 0=OFF, non-zero value=ON
wet_sw	Wet-side sound (sound with effect applied) ON/OFF state

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 Error Codes.)

Description

This function turns on or off the sound to which effects are applied and the sound to which effects are not applied, respectively, after sound generation is performed for each voice.

The volumes set by *l* and *r* in sceSasSetVolume() are effective for *dry_sw* and the volumes set by *wl* and *wr* are effective for *wet_sw*.

If the value is set to 1 (non-zero), the input is turned on, and if the value is set to 0, the input is turned off.

If *wet_sw* = OFF, the effect will be muted.

In multichannel mode, this function returns an error without setting the parameter.

Examples

```
SceSasResult result;

/* Set Dry = ON and Wet = OFF */
result = sceSasSetEffect(1, 0);
if (result < 0) {
        printf("Error: sceSasSetEffect() %X\n", result);
}
```

Constants

Error Codes

libsas Error Codes

Definition

Macro	Value	Description
SCE_SAS_ERROR_NSMPL	0x80420001	Granularity specification is invalid
SCE_SAS_ERROR_MFMT	0x80420003	Output mode specification is invalid
SCE_SAS_ERROR_ADDRESS	0x80420005	Address alignment is invalid
SCE_SAS_ERROR_VOICE_INDEX	0x80420010	Voice number is invalid
SCE_SAS_ERROR_NOISE_CLOCK	0x80420011	Noise clock is invalid
SCE_SAS_ERROR_PITCH_VAL	0x80420012	Pitch specification is invalid
SCE_SAS_ERROR_ADSR_MODE	0x80420013	ADSR mode is invalid
SCE_SAS_ERROR_ADPCM_SIZE	0x80420014	ADPCM waveform data size is invalid
SCE_SAS_ERROR_LOOP_MODE	0x80420015	Loop mode or loop position specification is invalid
SCE_SAS_ERROR_INVALID_STATE	0x80420016	Cannot be executed in the current state (keyed-on, keyed-off, paused, etc.)
SCE_SAS_ERROR_VOLUME_VAL	0x80420018	Volume value for the voice is invalid
SCE_SAS_ERROR_ADSR_VAL	0x80420019	Either the ADSR value or the Sustain level value is invalid
SCE_SAS_ERROR_PCM_SIZE	0x8042001a	PCM size specification is invalid
SCE_SAS_ERROR_FX_TYPE	0x80420020	Effect type is invalid
SCE_SAS_ERROR_FX_FEEDBACK	0x80420021	Effect feedback value is invalid
SCE_SAS_ERROR_FX_DELAY	0x80420022	Effect delay value is invalid
SCE_SAS_ERROR_FX_VOLUME_VAL	0x80420023	Effect volume value is invalid
SCE_SAS_ERROR_FX_UNAVAILABLE	0x80420024	Effects setting function is unavailable in multichannel mode
SCE_SAS_ERROR_BUSY	0x80420030	`sceSasCore()` was called multiple times
SCE_SAS_ERROR_NO_CONCATENATE_SPACE	0x80420042	Not in a state in which data can be replenished
SCE_SAS_ERROR_NOTINIT	0x80420100	Initialization has not been performed
SCE_SAS_ERROR_ALRDYINIT	0x80420101	Initialization has already been performed