libaudioenc Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

Datatypes

SceAudioencInitParam

Union for libaudioenc initialization

Definition

```
#include <audioenc.h>
typedef union SceAudioencInitParam {
        SceUInt32 size;
        SceAudioencInitStreamParam celp;
} SceAudioencInitParam;
```

Members

size	Size of the structure corresponding to the type of audio encoder to be used
celp	CELP initialization structure

Description

This is the union for libaudioenc initialization.

This union is used to initialize libaudioenc using `sceAudioencInitLibrary()`.

To *size*, do not specify `sizeof(SceAudioencInitParam)`. Instead, specify the size of the structure corresponding to the type of audio encoder to be used.

See Also

`SceAudioencInitStreamParam`, `sceAudioencInitLibrary()`

SceAudioencInitStreamParam

Structure for libaudioenc stream initialization

Definition

```
#include <audioenc.h>
typedef struct SceAudioencInitStreamParam {
        SceUInt32 size;
        SceUInt32 totalStreams;
} SceAudioencInitStreamParam;
```

Members

size	Size of the structure
totalStreams	Number of streams available for encoding at the same time

Description

This is the structure for libaudioenc stream initialization.

This structure is used to initialize libaudioenc CELP encoders.

Note that *totalStreams* has an upper limit. *totalStreams* should not be set higher than the maximum value for the total number of streams available for encoding at the same time.

For example, in the CELP encoder, the maximum value for the number of streams, `SCE_AUDIOENC_CELP_MAX_STREAMS` is `1`, therefore specify 1 to *totalStreams*.

See Also

`SceAudioencInitParam`, `sceAudioencInitLibrary()`, Maximum Value for the Number of Streams Available for Encoding at the Same Time

SceAudioencCtrl

Audio encoder control structure

Definition

```
#include <audioenc.h>
typedef struct SceAudioencCtrl {
        SceUInt32 size;
        SceInt32 handle;
        SceUInt8 *pInputPcm;
        SceUInt32 inputPcmSize;
        SceUInt32 maxPcmSize;
        void *pOutputEs;
        SceUInt32 outputEsSize;
        SceUInt32 maxEsSize;
        SceUInt32 wordLength;
        SceAudioencInfo *pInfo;
        SceAudioencOptInfo *pOptInfo;
} SceAudioencCtrl;
```

Members

size	Size of the structure
handle	Encoder handle
pInputPcm	Pointer to input PCM buffer
inputPcmSize	Size of input PCM used (in Bytes)
maxPcmSize	Maximum size of PCM being used (in Bytes)
pOutputEs	Pointer to output elementary stream buffer
outputEsSize	Size of output elementary stream (in Bytes)
maxEsSize	Maximum size of elementary stream to be output (in Bytes)
wordLength	Number of PCM quantization bits
pInfo	Pointer to audio encoder information structure
pOptInfo	Pointer to optional information structure (provided for future expansion)

Description

This structure is used to control audio encoders.

By calling `sceAudioencCreateEncoder()` using this structure, the encoder handle will be set and the structure and audio encoder will be associated. Thereafter, associated audio encoders can be used by calling various functions through this structure. At the end, release the association between this structure and audio encoders by calling `sceAudioencDeleteEncoder()` through this structure.

Refer to each function regarding parameters that need to be set when calling libaudioenc functions.

See Also

`SceAudioencInfo`, `sceAudioencCreateEncoder()`, `sceAudioencDeleteEncoder()`, `sceAudioencEncode()`, Number of PCM Quantization Bits

SceAudioencInfo

Audio encoder information union

Definition

```
#include <audioenc.h>
typedef union SceAudioencInfo {
        SceUInt32 size;
        SceAudioencInfoCelp celp;
} SceAudioencInfo;
```

Members

size	Size of the structure corresponding to the type of audio encoder to be used
celp	CELP information structure

Description

This union is used to set and obtain audio encoder information.

Refer to each function regarding parameters that need to be set when calling libaudioenc functions.

To *size*, do not specify `sizeof(SceAudioencInfo)`. Instead, specify the size of the structure corresponding to the type of audio encoder to be used.

See Also

`SceAudioencCtrl`, `SceAudioencInfoCelp`, `sceAudioencCreateEncoder()`, `sceAudioencDeleteEncoder()`

SceAudioencInfoCelp

CELP information structure

Definition

```
#include <audioenc.h>
typedef struct SceAudioencInfoCelp {
        SceUInt32 size;
        SceUInt32 excitationMode;
        SceUInt32 samplingRate;
        SceUInt32 bitRate;
} SceAudioencInfoCelp;
```

Members

size	Size of the structure
excitationMode	Excitation mode
samplingRate	Sampling frequency (in Hz)
bitRate	Bit rate (in bps)

Description

This structure is for CELP information.

See Also

`SceAudioencInfo`, `sceAudioencCreateEncoder()`, `sceAudioencDeleteEncoder()`

SceAudioencOptInfo

Audio encoder optional information union

Definition

```
#include <audioenc.h>
typedef union SceAudioencOptInfo {
        SceUInt32 size;
        SceAudioencOptInfoCelp celp;
} SceAudioencOptInfo;
```

Members

size	Size of the structure corresponding to the type of audio encoder to be used
celp	CELP optional information structure

Description

This union is used to set and obtain audio encoder optional information.

Refer to each function regarding parameters that need to be set when calling libaudioenc functions.

To *size*, do not specify `sizeof(SceAudioencOptInfo)`. Instead, specify the size of the structure corresponding to the type of audio encoder to be used.

See Also

`SceAudioencCtrl`, `SceAudioencOptInfoCelp`, `sceAudioencGetOptInfo()`

SceAudioencOptInfoCelp

CELP optional information structure

Definition

```
#include <audioenc.h>
typedef struct SceAudioencOptInfoCelp {
        SceUInt32 size;
        SceUInt8 header[32];
        SceUInt32 headerSize;
        SceUInt32 encoderVersion;
} SceAudioencOptInfoCelp;
```

Members

size	Size of the structure
header	header information
headerSize	header size (in bytes)
encoderVersion	encoder version

Description

This structure is for CELP optional information.

See Also

`SceAudioencOptInfo`, `sceAudioencGetOptInfo()`

Initializing / Terminating the Library

sceAudioencInitLibrary

Initialize libaudioenc

Definition

```
#include <audioenc.h>
SceInt32 sceAudioencInitLibrary (
        SceUInt32 codecType,
        SceAudioencInitParam *pInitParam
)
```

Arguments

codecType	Type of audio encoder
pInitParam	Pointer to libaudioenc initialization structure

Return Values

Value	Description
0 (`SCE_OK`)	Success

<0	Error
	`SCE_AUDIOENC_ERROR_API_FAIL`
	`SCE_AUDIOENC_ERROR_INVALID_TYPE`
	`SCE_AUDIOENC_ERROR_INVALID_INIT_PARAM`
	`SCE_AUDIOENC_ERROR_ALREADY_INITILIZED`
	`SCE_AUDIOENC_ERROR_OUT_OF_MEMORY`

Description

This function is used to initialize libaudioenc.

To *pInitParam*, specify the pointer to the libaudioenc initialization structure with initialization parameters set for each corresponding type of audio encoder. By calling this function, the required amount of memory will be allocated from the Codec Engine memory, and libaudioenc will be initialized. To release the allocated memory, call `sceAudioencTermLibrary()`.

Notes

This function is multi-thread safe.

Examples

```
SceAudioencInitParam audioencInitParam;

// Sets audio encoder initialization parameters concerning CELP library
memset(&audioencInitParam, 0, sizeof(audioencInitParam));
audioencInitParam.size = sizeof(audioencInitParam.celp);
audioencInitParam.celp.totalStreams = 1;

// Initializes the CELP library
res = sceAudioencInitLibrary(SCE_AUDIOENC_TYPE_CELP, &audioencInitParam);
if (res < 0) {
        //Error handling
}
```

See Also

`SceAudioencInitParam`, `SceAudioencInitStreamParam`, `sceAudioencTermLibrary()`

sceAudioencTermLibrary

Terminate libaudioenc

Definition

```
#include <audioenc.h>
SceInt32 sceAudioencTermLibrary (
        SceUInt32 codecType
)
```

Arguments

codecType	Type of audio encoder

Return Values

Value	Description
0 (SCE_OK)	Success
<0	Error SCE_AUDIOENC_ERROR_INVALID_TYPE SCE_AUDIOENC_ERROR_NOT_INITIALIZED SCE_AUDIOENC_ERROR_A_HANDLE_IN_USE

Description

This function is used to terminate libaudioenc.

Call this function to delete all generated audio encoders and terminate libaudioenc. By calling this function, the memory area allocated by `sceAudioencInitLibrary()` will be released. Note that when this function is called, all audio encoders corresponding to the specified type of audio encoder need to be deleted.

Notes

This function is multi-thread safe.

Examples

```
// Terminates the CELP library
res = sceAudioencTermLibrary(SCE_AUDIOENC_TYPE_CELP);
if (res < 0) {
        //Error handling
}
```

See Also

`sceAudioencInitLibrary()`

Generating / Deleting Audio Encoders

sceAudioencCreateEncoder

Generate audio encoders

Definition

```
#include <audioenc.h>
SceInt32 sceAudioencCreateEncoder (
        SceAudioencCtrl *pCtrl,
        SceUInt32 codecType
)
```

Arguments

pCtrl	Pointer to audio encoder control structure
codecType	Type of audio encoder

Return Values

Value	Description
0(SCE_OK)	Success
<0	Error SCE_AUDIOENC_ERROR_API_FAIL SCE_AUDIOENC_ERROR_INVALID_TYPE SCE_AUDIOENC_ERROR_NOT_INITIALIZED SCE_AUDIOENC_ERROR_ALL_HANDLES_IN_USE SCE_AUDIOENC_ERROR_INVALID_PTR SCE_AUDIOENC_ERROR_CH_SHORTAGE SCE_AUDIOENC_ERROR_INVALID_WORD_LENGTH SCE_AUDIOENC_ERROR_INVALID_SIZE SCE_AUDIOENC_CELP_ERROR_INVALID_CONFIG

Description

This function generates audio encoders.

By calling this function, the memory secured with `sceAudioencInitLibrary()` will be allocated to the generated audio encoders.

Parameters set in `SceAudioencCtrl` will depend on the type of audio encoder. Refer to Table 1 and Table 2 for parameter settings when calling this function.

Notes

This function is multi-thread safe.

Table 1　SceAudioencCtrl structure when calling sceAudioencCreateEncoder()

Member variable in `SceAudioencCtrl` structure	CELP	
	in	out
size	○	
handle		○
pInputPcm		
inputPcmSize		
maxPcmSize		○
pOutputEs		
outputEsSize		
maxEsSize		○
wordLength	○	
pInfo	○	
pOptInfo		

Table 2　SceAudioencInfo structure when calling sceAudioencCreateEncoder()

Member variable in `SceAudioencInfoCelp` structure	CELP	
	in	out
size	○	
excitationMode	○	
samplingRate	○	
bitRate	○	

Examples

```
SceAudioencCtrl audioencCtrl;
SceAudioencInfo audioencInfo;

// Set SceAudioencInfo
memset(&audioencInfo, 0, sizeof(SceAudioencInfo));
audioencInfo.size = sizeof(audioencInfo.celp);

// Set SceAudioencCtrl
memset(&audioencCtrl, 0, sizeof(SceAudioencCtrl));
audioencCtrl.size = sizeof(SceAudioencCtrl);

// Set CELP stream data
audioencCtrl.wordLength = SCE_AUDIOENC_WORD_LENGTH_16BITS;
audioencInfo.celp.excitationMode = SCE_AUDIOENC_CELP_MPE;
audioencInfo.celp.samplingRate = SCE_AUDIOENC_CELP_SAMPLING_RATE_8KHZ;
audioencInfo.celp.bitRate = SCE_AUDIOENC_CELP_BIT_RATE_3850BPS;
audioenc.pInfo = &audioencInfo;

// Generate CELP audio encorders
res = sceAudioencCreateEncoder(&audioencCtrl, SCE_AUDIOENC_TYPE_CELP);
if (res < 0) {
        //Error handling
}
```

See Also

`SceAudioencCtrl`, `SceAudioencInfo`, `sceAudioencDeleteEncoder()`

sceAudioencDeleteEncoder

Delete audio encoders

Definition

```
#include <audioenc.h>
SceInt32 sceAudioencDeleteEncoder (
        SceAudioencCtrl *pCtrl
)
```

Arguments

pCtrl	Pointer to the audio encoder control structure

Return Values

Value	Description
0 (SCE_OK)	Success
<0	Error SCE_AUDIOENC_ERROR_API_FAIL SCE_AUDIOENC_ERROR_INVALID_TYPE SCE_AUDIOENC_ERROR_NOT_INITIALIZED SCE_AUDIOENC_ERROR_INVALID_PTR SCE_AUDIOENC_ERROR_INVALID_HANDLE SCE_AUDIOENC_ERROR_NOT_HANDLE_IN_USE SCE_AUDIOENC_ERROR_INVALID_WORD_LENGTH SCE_AUDIOENC_ERROR_INVALID_SIZE

Description

This function deletes audio encoders.

By calling this function, the memory allocated for the audio encoders using `sceAudioencCreateEncoder()` will be released. When terminating libaudioenc by using `sceAudioencTermLibrary()`, all audio encoders corresponding to the type of audio encoder need to be deleted by using this function.

Parameters set in `SceAudioencCtrl` will depend on the type of audio encoder. Refer to Table 3 and Table 4 for parameter settings when calling this function.

Notes

This function is multi-thread safe.

Table 3 SceAudioencCtrl structure when calling sceAudioencDeleteEncoder()

Member variable in `SceAudioencCtrl` structure	CELP	
	in	out
size	○	
handle	○	
pInputPcm		
inputPcmSize		
maxPcmSize		
pOutputEs		
outputEsSize		
maxEsSize		
wordLength	○	
pInfo	○	
pOptInfo		

Table 4 SceAudioencInfo structure when calling sceAudioencDeleteEncoder()

Member variable in `SceAudioencInfoCelp` structure	CELP	
	in	out
size	○	
excitationMode		
samplingRate		
bitRate		

Examples

```
SceAudioencCtrl audioencCtrl;

// Generate audio encoders

// Delete audio encoders
res = sceAudioencDeleteEncoder (&audioencCtrl);
if (res < 0) {
        //Error handling
}
```

See Also

`SceAudioencCtrl`, `sceAudioencCreateEncoder()`

Encoding Audio Data

sceAudioencEncode

Encode audio data

Definition

```
#include <audioenc.h>
SceInt32 sceAudioencEncode (
        SceAudioencCtrl *pCtrl
)
```

Arguments

pCtrl	Pointer to the audio encoder control structure

Return Values

Value	Description
0 (`SCE_OK`)	Success

<0	Error
	`SCE_AUDIOENC_ERROR_API_FAIL`
	`SCE_AUDIOENC_ERROR_INVALID_TYPE`
	`SCE_AUDIOENC_ERROR_NOT_INITIALIZED`
	`SCE_AUDIOENC_ERROR_INVALID_PTR`
	`SCE_AUDIOENC_ERROR_INVALID_HANDLE`
	`SCE_AUDIOENC_ERROR_NOT_HANDLE_IN_USE`
	`SCE_AUDIOENC_ERROR_INVALID_WORD_LENGTH`
	`SCE_AUDIOENC_ERROR_INVALID_SIZE`

Description

This function encodes audio data.

By calling this function, input PCM data loaded to *pInputPcm* will be encoded, and encoded elementary streams data in *pOutputEs* will be overwritten. At this time, the input PCM size used for encoding and the output elementary stream size are stored to *inputPcmSize* and *outputEsSize*.

Parameters set in `SceAudioencCtrl` will depend on the type of audio encoder. Refer to Table 5 and Table 6 for parameter settings when calling this function.

Notes

- The maximum value of *inputPcmSize* will be set in *maxPcmSize* when `sceAudioencCreateEncoder()` is called. For the buffer set in *pInputPcm*, set aside an area equal to or greater than *maxPcmSize*.

- The *pInputPcm* buffer area will be accessed by both the ARM and the Codec Engine. At this time, cache coherency must be secured between the ARM and the Codec Engine.
  In order to secure this cache coherency, **memory which is 256 bytes aligned and whose size is a multiple of 256 bytes must be allocated for the *pInputPcm* buffer. However, *pInputPcm*, the starting address of the PCM data, does not require 256 bytes of alignment. Do not specify the same buffer area at the same time for several encoders.**

- The maximum value of *outputEsSize* will be set in *maxEsSize* when `sceAudioencCreateEncoder()` is called. For the buffer set in *pOutputEs*, set aside an area equal to or greater than *maxEsSize*.

- The *pOutputEs* buffer area will be accessed by both the ARM and the Codec Engine. At this time, cache coherency must be secured between the ARM and the Codec Engine.
  In order to secure this cache coherency, **memory which is 256 bytes aligned and whose size is a multiple of 256 bytes must be allocated for the *pOutputEs* buffer. However, *pOutputEs*, the starting address of the elementary stream, does not require 256 bytes of alignment.**

Notes

This function is not multi-thread safe for the same encoder.

Table 5 SceAudioencCtrl structure when calling sceAudioencEncode()

Member variable in `SceAudioencCtrl` structure	CELP	
	in	out
size	○	
handle	○	
pInputPcm	○	
inputPcmSize		○
maxPcmSize		
pOutputEs	○	
outputEsSize		○
maxEsSize		
wordLength	○	
pInfo	○	
pOptInfo		

Table 6 SceAudioencInfo structure when calling sceAudioencEncode()

Member variable in `SceAudioencInfoCelp` structure	CELP	
	in	out
size	○	
excitationMode		
samplingRate		
bitRate		

Examples

```
SceAudioencCtrl audioencCtrl;

// Generate audio encoders

// Set input/output buffer
audioencCtrl.pInputPcm = inputPcmBuffer;
audioencCtrl.pOutputEs = outputEsBuffer;

// Encode audio data
res = sceAudioencEncode(&audioencCtrl);
if (res < 0) {
        //Error handling
}
```

See Also

`SceAudioencCtrl`

sceAudioencClearContext

Reinitialize audio encoders

Definition

```
#include <audioenc.h>
SceInt32 sceAudioencClearContext (
        SceAudioencCtrl *pCtrl
)
```

Arguments

pCtrl	Pointer to the audio encoder control structure

Return Values

Value	Description
0 (`SCE_OK`)	Success

<0	Error
	`SCE_AUDIOENC_ERROR_API_FAIL`
	`SCE_AUDIOENC_ERROR_INVALID_TYPE`
	`SCE_AUDIOENC_ERROR_NOT_INITIALIZED`
	`SCE_AUDIOENC_ERROR_INVALID_PTR`
	`SCE_AUDIOENC_ERROR_INVALID_HANDLE`
	`SCE_AUDIOENC_ERROR_NOT_HANDLE_IN_USE`
	`SCE_AUDIOENC_ERROR_INVALID_WORD_LENGTH`
	`SCE_AUDIOENC_ERROR_INVALID_SIZE`

Description

This function reinitializes audio encoders.

By calling this function, the context memory is cleared and audio encoders are reinitialized.

Parameters set in `SceAudioencCtrl` depend on the type of audio encoder. Refer to Table 7 and Table 8 for parameter settings when calling this function.

This function is used for encoding non-continuous audio data.

Notes

This function is not multi-thread safe for the same encoder.

Table 7 SceAudioencCtrl structure when calling sceAudioencClearContext()

Member variable in `SceAudioencCtrl` structure	CELP in	CELP out
size	○	
handle	○	
pInputPcm		
inputPcmSize		
maxPcmSize		
pOutputEs		
outputEsSize		
maxEsSize		
wordLength	○	
pInfo	○	
pOptInfo		

Table 8 SceAudioencInfo structure when calling sceAudioencClearContext()

Member variable in `SceAudioencInfoCelp` structure	CELP in	CELP out
size	○	
excitationMode	○	
samplingRate	○	
bitRate	○	

Examples

```
SceAudioencCtrl audioencCtrl;

// Generate audio encoders

// Encode audio data

// Reinitialize audio encoders
res = sceAudioencClearContext(&audioencCtrl);
if (res < 0) {
        //Error handling
}
```

See Also

`SceAudioencCtrl`

Obtaining Information

sceAudioencGetOptInfo

Obtain optional information

Definition

```
#include <audioenc.h>
SceInt32 sceAudioencGetOptInfo (
        SceAudioencCtrl *pCtrl
)
```

Arguments

pCtrl	Pointer to the audio encoder control structure

Return Values

Value	Description
0 (SCE_OK)	Success
<0	Error SCE_AUDIOENC_ERROR_INVALID_TYPE SCE_AUDIOENC_ERROR_NOT_INITIALIZED SCE_AUDIOENC_ERROR_INVALID_PTR SCE_AUDIOENC_ERROR_INVALID_HANDLE SCE_AUDIOENC_ERROR_NOT_HANDLE_IN_USE SCE_AUDIOENC_ERROR_INVALID_WORD_LENGTH SCE_AUDIOENC_ERROR_INVALID_SIZE

Description

This function obtains optional information from audio encoders.

By calling this function, optional information can be obtained regarding audio encoder.

Parameters set in SceAudioencCtrl depend on the type of audio encoder. Refer to Table 9, Table 10 and Table 11 for parameter settings when calling this function.

This function is provided for supporting debugging. Programming that uses data obtained with this function to modify controls is not recommended.

Notes

This function is not multi-thread safe for the same encoder.

Table 9 SceAudioencCtrl structure when calling sceAudioencGetOptInfo()

Member variable in `SceAudioencCtrl` structure	CELP	
	in	out
size	○	
handle	○	
pInputPcm		
inputPcmSize		
maxPcmSize		
pOutputEs		
outputEsSize		
maxEsSize		
wordLength	○	
pInfo	○	
pOptInfo	○	

Table 10 SceAudioencInfo structure when calling sceAudioencGetOptInfo()

Member variable in `SceAudioencInfoCelp` structure	CELP	
	in	out
size	○	
excitationMode		
samplingRate		
bitRate		

Table 11 SceAudioencInfo structure when calling sceAudioencGetOptInfo()

Member variable in `SceAudioencOptInfoCelp` structure	CELP	
	in	out
size	○	
header		○
headerSize		○
encoderVersion		○

Examples

```
SceAudioencCtrl audioencCtrl;
SceAudioencOptInfo audioencOptInfo;

// Generate audio encoders

// Set SceAudioencOptInfo
audioencOptInfo.size = sizeof(audioencOptInfo.celp);
audioencCtrl.pOptInfo = &audioencOptInfo;

// Obtain optional information from audio encoders
res = sceAudioencGetOptInfo(&audioencCtrl);
if (res < 0) {
```

```
                //Error handling
        }
```

See Also

`SceAudioencCtrl`, `SceAudioencOptInfo`, `SceAudioencOptInfoCelp`

sceAudioencGetInternalError

Obtain internal errors

Definition

```
#include <audioenc.h>
SceInt32 sceAudioencGetInternalError (
        SceAudioencCtrl *pCtrl,
        SceInt32 *pInternalError
)
```

Arguments

pCtrl	Pointer to the audio encoder control structure
pInternalError	Pointer to internal error variables

Return Values

Value	Description
0 (SCE_OK)	Success
<0	Error SCE_AUDIOENC_ERROR_INVALID_TYPE SCE_AUDIOENC_ERROR_NOT_INITIALIZED SCE_AUDIOENC_ERROR_INVALID_PTR SCE_AUDIOENC_ERROR_INVALID_HANDLE SCE_AUDIOENC_ERROR_NOT_HANDLE_IN_USE SCE_AUDIOENC_ERROR_INVALID_WORD_LENGTH SCE_AUDIOENC_ERROR_INVALID_SIZE

Description

This function obtains internal errors from audio encoders.

By calling this function, details can be obtained regarding SCE_AUDIOENC_ERROR_API_FAIL internal errors within the Codec Engine.

Parameters set in SceAudioencCtrl depend on the type of audio encoder. Refer to Table 12 and Table 13 for parameter settings when calling this function.

This function is provided for supporting debugging. Programming that uses data obtained with this function to modify controls is not recommended.

Notes

This function is not multi-thread safe for the same encoder.

Table 12 SceAudioencCtrl structure when calling sceAudioencGetInternalError()

Member variable in `SceAudioencCtrl` structure	CELP	
	in	out
size	○	
handle	○	
pInputPcm		
inputPcmSize		
maxPcmSize		
pOutputEs		
outputEsSize		
maxEsSize		
wordLength	○	
pInfo	○	
pOptInfo		

Table 13 SceAudioencInfo structure when calling sceAudioencGetInternalError()

Member variable in `SceAudioencInfoCelp` structure	CELP	
	in	out
size	○	
excitationMode		
samplingRate		
bitRate		

Examples

```
SceAudioencCtrl audioencCtrl;
SceInt32 internalError;

// Generate audio encoders

// Obtain internal errors from audio encoders
res = sceAudioencGetInternalError(&audioencCtrl, &internalError);
if (res < 0) {
        //Error handling
}
```

See Also

`sceAudioencCreateEncoder()`, `sceAudioencDeleteEncoder()`, `sceAudioencEncode()`, `sceAudioencClearContext()`

Constants

Audio Encoder Types

Audio encoder types

Definition

Value	(Number)	Description
`SCE_AUDIOENC_TYPE_CELP`	0x2006U	CELP

Description

This is an identifier that indicates the type of audio encoder.

When calling `sceAudioencInitLibrary()`, `sceAudioencTermLibrary()`, or `sceAudioencCreateEncoder()`, specify this identifier.

Maximum Value for the Number of Streams Available for Encoding at the Same Time

Maximum value for the number of streams available for encoding at the same time

Definition

Value	(Number)	Description
`SCE_AUDIOENC_CELP_MAX_STREAMS`	1	Maximum value for the number of CELP streams available for encoding at the same time using libaudioenc

Description

This identifier indicates the maximum value for the number of streams that can be encoded by libaudioenc at the same time.

When specifying the *`totalStreams`* variable in the `SceAudioencInitStreamParam` structure, ensure that it does not exceed this value.

Number of PCM Quantization Bits

Number of PCM quantization bits

Definition

Value	(Number)	Description
`SCE_AUDIOENC_WORD_LENGTH_16BITS`	16	16 bits

Description

This identifier indicates the number of PCM quantization bits for audio encoders.

Set this identifier to the *`wordLength`* variable in the `SceAudioencCtrl` structure.

Maximum Number of Input Samples

Maximum number of Input samples

Definition

Value	(Number)	Description
`SCE_AUDIOENC_CELP_MAX_SAMPLES`	320	Maximum number of input samples for CELP encoders

Description

This identifier indicates the maximum number of input samples for audio encoder.

Each time `sceAudioencEncode()` is called, the encoding PCM data is input up to a maximum number of samples as the value of this identifier.

Maximum Size of Elementary Streams

Maximum size of elementary streams

Definition

Value	(Number)	Description
`SCE_AUDIOENC_CELP_MAX_ES_SIZE`	27	Maximum size of elementary streams for CELP encoders

Description

This identifier indicates the maximum size of elementary streams for audio encoders.

Each time `sceAudioencEncode()` is called, elementary streams will be encoded up to a maximum of the value of this identifier.

CELP Excitation Mode

CELP excitation mode

Definition

Value	(Number)	Description
`SCE_AUDIOENC_CELP_MPE`	0	Multi-pulse excitation

Description

This identifier indicates the CELP excitation mode.

CELP Sampling Rate

CELP sampling rate

Definition

Value	(Number)	Description
`SCE_AUDIOENC_CELP_SAMPLING_RATE_8KHZ`	8000	8 kHz

Description

This identifier indicates the CELP sampling rate.

CELP Bit Rate

CELP bit rate

Definition

Value	(Number)	Description
SCE_AUDIOENC_CELP_BIT_RATE_3850BPS	3850	3850 bps
SCE_AUDIOENC_CELP_BIT_RATE_4650BPS	4650	4650 bps
SCE_AUDIOENC_CELP_BIT_RATE_5700BPS	5700	5700 bps
SCE_AUDIOENC_CELP_BIT_RATE_6600BPS	6600	6600 bps
SCE_AUDIOENC_CELP_BIT_RATE_7300BPS	7300	7300 bps
SCE_AUDIOENC_CELP_BIT_RATE_8700BPS	8700	8700 bps
SCE_AUDIOENC_CELP_BIT_RATE_9900BPS	9900	9900 bps
SCE_AUDIOENC_CELP_BIT_RATE_10700BPS	10700	10700 bps
SCE_AUDIOENC_CELP_BIT_RATE_11800BPS	11800	11800 bps
SCE_AUDIOENC_CELP_BIT_RATE_12200BPS	12200	12200 bps

Description

This identifier indicates the CELP bit rate.

Error Codes

List of error codes returned by libaudioenc

Definition

Value	(Number)	Description
SCE_AUDIOENC_ERROR_API_FAIL	0x80860000	An internal error has occurred in the Codec Engine
SCE_AUDIOENC_ERROR_INVALID_TYPE	0x80860001	Audio encoder type is invalid
SCE_AUDIOENC_ERROR_INVALID_INIT_PARAM	0x80860002	Initialization parameter of libaudioenc is invalid
SCE_AUDIOENC_ERROR_ALREADY_INITIALIZED	0x80860003	libaudioenc has already been initialized
SCE_AUDIOENC_ERROR_OUT_OF_MEMORY	0x80860004	Insufficient memory
SCE_AUDIOENC_ERROR_NOT_INITIALIZED	0x80860005	libaudioenc has not been initialized
SCE_AUDIOENC_ERROR_A_HANDLE_IN_USE	0x80860006	A encoder is currently being used
SCE_AUDIOENC_ERROR_ALL_HANDLES_IN_USE	0x80860007	All handles are being used
SCE_AUDIOENC_ERROR_INVALID_PTR	0x80860008	The specified pointer is invalid
SCE_AUDIOENC_ERROR_INVALID_HANDLE	0x80860009	The `SceAudioencCtrl` structure handle is invalid
SCE_AUDIOENC_ERROR_NOT_HANDLE_IN_USE	0x8086000A	The `SceAudioencCtrl` structure handle has not been used
SCE_AUDIOENC_ERROR_CH_SHORTAGE	0x8086000B	Insufficient number of channels available for encoding at the same time
SCE_AUDIOENC_ERROR_INVALID_WORD_LENGTH	0x8086000C	The number of PCM quantization bits in the `SceAudioencCtrl` structure is invalid
SCE_AUDIOENC_ERROR_INVALID_SIZE	0x8086000D	The size of the `SceAudioencCtrl` structure is invalid
SCE_AUDIOENC_ERROR_INVALID_ALIGNMENT	0x8086000E	The specified pointer alignment is invalid
SCE_AUDIOENC_ERROR_UNSUPPORTED	0x8086000F	The executed function is not supported
SCE_AUDIOENC_CELP_ERROR_INVALID_CONFIG	0x80861001	CELP information structure settings are invalid