libaudioenc Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

1 Library Overview

Scope of This Document

This document describes libaudioenc, which encodes PCM audio data. This describes the basic procedure for encoding and the restrictions regarding the I/O buffers used during encoding.

Purpose and Features

libaudioenc is a library for encoding PCM-format audio data with a Codec Engine. Applications can use libaudioenc to encode PCM data into audio data of specific codec type.

libaudioenc supports the following audio codec.

CELP

Supported channels	: 1 channel
Supported sampling frequencies	: 8000Hz
Supported bit rates	: 3850/ 4650/ 5700/ 6600/ 7300/ 8700/ 9900/ 10700/ 11800/ 12200 bps

Embedding in Program

The following files are required in order to use libaudioenc.

Filename	Description
audioenc.h	Header file
libSceAudioenc_stub.a	Stub library file

Include audioenc.h in source program. When building the program, link libSceAudioenc_stub.a.

Sample Programs

The following program is provided as a libaudioenc sample program for reference purposes.

sample_code/audio_video/api_libaudioenc/celp_realtime/

This sample shows the basic usage of libaudioenc.

Reference Materials

Refer to the following document for audio input.

- "Audio Input Function Overview"

2 Usage

Basic Usage Procedure

This chapter describes the basic procedure for audio encode processing.

(1) Initialize audio codec library to be used

libaudioenc needs to be initialized for each audio codec to be used. Set the initialization parameters of the audio codecs to be used to the variables of the `SceAudioencInitParam` type. Specifying the variables of the `SceAudioencInitParam` type to an argument and calling `sceAudioencInitLibrary()` initialize libaudioenc. Resources are allocated within the library at this time.

(2) Generate audio encoders

Set the parameters for generating the audio encoders to the variables of the `SceAudioencCtrl` type. Specifying the variables of the `SceAudioencCtrl` type to an argument and calling `sceAudioencCreateEncoder()` generate the audio encoders.

(3) Encode audio data

Set the pointers to the PCM buffer and the elementary stream buffer to the variables of the `SceAudioencCtrl` type. Specifying the variables of the `SceAudioencCtrl` type to an argument and calling `sceAudioencEncode()` encode the audio data.

(4) Delete audio encoders

When the generated audio encoders are no longer needed, calling `sceAudioencDeleteEncoder()` deletes the audio encoders.

(5) Terminate libaudioenc

When the initialized audio codecs are no longer needed, calling `sceAudioencTermLibrary()` executes libaudioenc terminating processing. This frees resources within the library that were allocated at initialization. All audio encoders of the terminated audio codecs must be deleted at this time.

Major APIs Used in Basic Processing

API	Description
`sceAudioencInitLibrary()`	Initializes libaudioenc
`sceAudioencTermLibrary()`	Terminates libaudioenc
`sceAudioencCreateEncoder()`	Generates audio encoders
`sceAudioencDeleteEncoder()`	Deletes audio encoders
`sceAudioencEncode()`	Encodes audio data
`sceAudioencClearContext()`	Clears the context of audio encoders and returns to the state after initialization

3 Notes

Input Buffer

ARM sets Data referred in the Codec Engine to the input buffer region of `sceAudioencEncode()`. To maintain cache coherency on the ARM and Codec Engine sides at this time, the following restrictions apply.

- The input buffer region must have a 256-byte alignment and its size must be a multiple of 256-byte.
- Allocate the maximum size (*`maxPcmSize`*) of the PCM data that is to be input from *`pInputPcm`* of `SceAudioencCtrl`, which is the input starting address, within the input buffer. However, there are no alignment restrictions for *`pInputPcm`*.
- Do not specify the same input buffer region for multiple encoders at the same time.

Output Buffer

Encoding result is written to the output buffer region of `sceAudioencEncode()` in the Codec Engine. To maintain cache coherency on the ARM and Codec Engine sides at this time, the following restrictions apply.

- The output buffer region must have a 256-byte alignment and its size must be a multiple of 256-byte.
- Allocate the maximum size (*`maxEsSize`*) of the elementary stream that is to be output from *`pOutputEs`* of `SceAudioencCtrl`, which is the output starting address, within the output buffer. However, there are no alignment restrictions for *`pOutputEs`*.

Input PCM Data

Only PCM data with a bit length of 16 bits can be input.