libaudiodec Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

1 Library Overview

Features

libaudiodec is a library for decoding audio data with a Codec Engine. Applications can use libaudiodec to decode various types of audio data to PCM data.

libaudiodec supports the following audio codecs.

ATRAC9™

Supported channels	: 1 channel, 2 channels
Supported sampling frequencies	: 12000/ 24000/ 48000 Hz
Supported bit rates	: 48/ 60/ 70/ 84/ 96/ 120/ 144/ 168/ 192 kbps

MP3

Supported codecs	: MPEG1/ 2/ 2.5 Layer3
Supported channels	: 1 channel, 2 channels
Supported sampling frequencies	: 8000/ 11025/ 12000/ 16000/ 22050/ 24000/ 32000/ 44100/ 48000 Hz
Supported bit rates	: 8/ 16/ 24/ 32/ 40/ 48/ 56/ 64/ 80/ 96/ 112/ 128/ 144/ 160/ 192/ 224/ 256/ 320 kbps

AAC

Supported codecs	: AAC-LC, HE-AAC v1/v2
Supported channels	: 1 channel, 2 channels
Supported sampling frequencies	: 8000/ 11025/ 12000/ 16000/ 22050/ 24000/ 32000/ 44100/ 48000 Hz
Supported bit rates	: 16 - 576 kbps

CELP

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

Files

The following files are required in order to use libaudiodec.

Filename	Description
audiodec.h	Header file
libSceAudiodec_stub.a	Stub library file

Sample Programs

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

sample_code/audio_video/api_libaudiodec/on_memory/

This sample shows the basic usage of libaudiodec.

sample_code/audio_video/tutorial_libaudiodec/at9dec_streaming/

This sample performs streaming playback of ATRAC9™ using libaudiodec.

Reference Materials

Refer to the following document for audio output.

- "Audio Output Function Overview"

2 Usage

Basic Usage Procedure

This chapter describes the basic procedure for audio decode processing.

(1) Initialize libaudiodec to be used

libaudiodec 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 `SceAudiodecInitParam` type. Specifying the variables of the `SceAudiodecInitParam` type to an argument and calling `sceAudiodecInitLibrary()` initialize libaudiodec. Resources are allocated within the library at this time.

(2) Generate audio decoders

Set the parameters for generating the audio decoders to the variables of the `SceAudiodecCtrl` type. Specifying the variables of the `SceAudiodecCtrl` type to an argument and calling `sceAudiodecCreateDecoder()` generate the audio decoders.

(3) Decode audio data

Set the pointers to the elementary stream buffer and the PCM buffer to the variables of the `SceAudiodecCtrl` type. Specifying the variables of the `SceAudiodecCtrl` type to an argument and calling `sceAudiodecDecode()` decode the audio data. At this time, `sceAudiodecDecodeNFrames()` or `sceAudiodecDecodeNStreams()` can be used instead of `sceAudiodecDecode()`.

(4) Delete audio decoders

When the generated audio decoders are no longer needed, calling `sceAudiodecDeleteDecoder()` deletes the audio decoders.

(5) Terminate libaudiodec

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

Major APIs Used in Basic Processing

API	Description
`sceAudiodecInitLibrary()`	Initializes libaudiodec
`sceAudiodecTermLibrary()`	Terminates libaudiodec
`sceAudiodecCreateDecoder()`	Generates audio decoders
`sceAudiodecDeleteDecoder()`	Deletes audio decoders
`sceAudiodecDecode()`	Decodes audio data
`sceAudiodecDecodeNFrames()`	Collectively decodes multiple frames
`sceAudiodecDecodeNStreams()`	Collectively decodes multiple streams
`sceAudiodecClearContext()`	Clears the context of audio decoders and returns to the state after initialization

3 Reference Information

MP3 File Tags

Music information called a **tag** is often added to an MP3 file. Information such as the artist's name or album name can be recorded in a tag. There are various types of tags, and the amount of information that can be maintained differs according to the type.

libaudiodec is not able to skip tags. Therefore, **MP3 data without the tags must be read into the input buffer.**

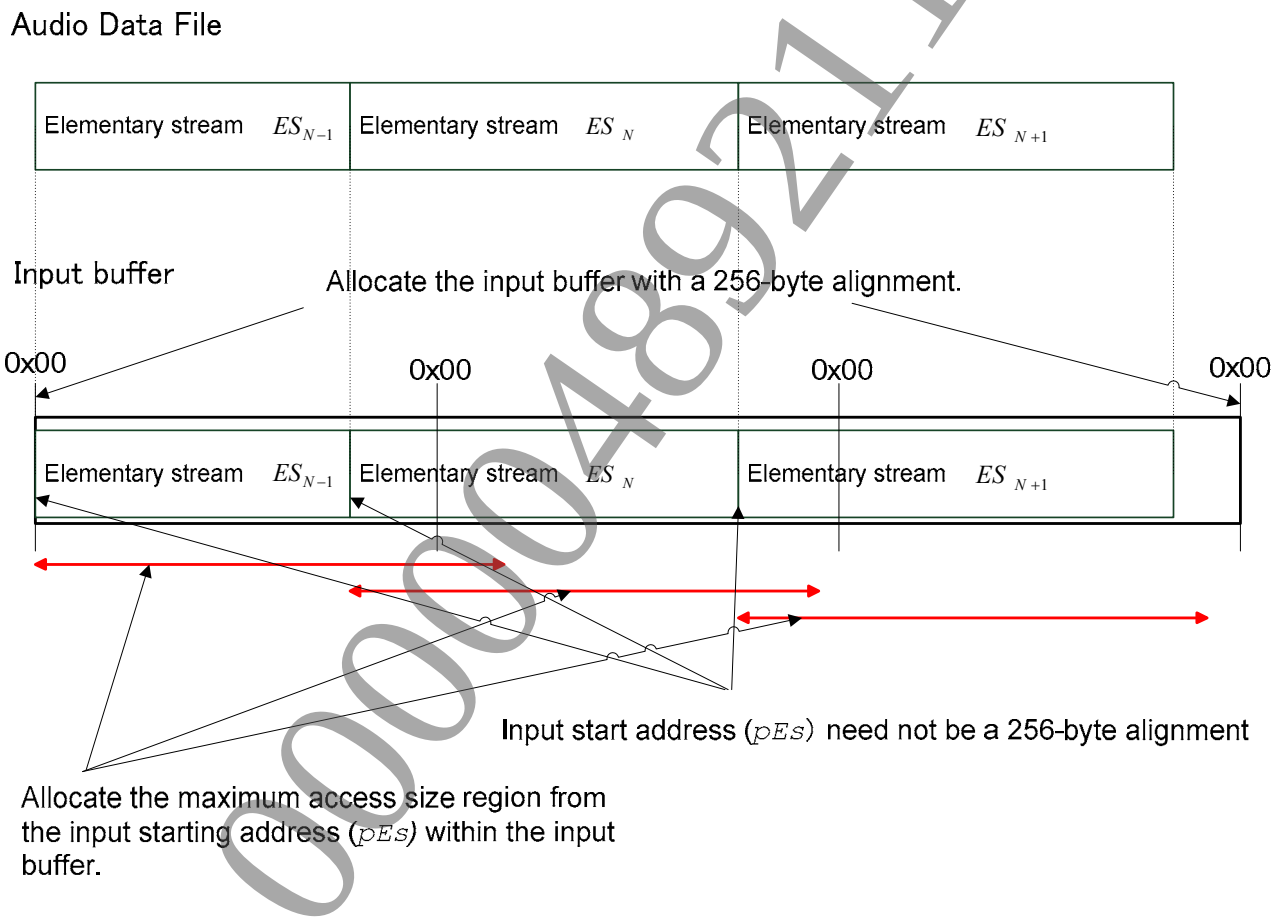
4 Notes

Input Buffer

ARM sets Data referred in the Codec Engine to the input buffer region of `sceAudiodecDecode()`. To maintain cache coherency on the ARM and Codec Engine sides at this time, the following restrictions apply. Also refer to Figure 1.

- Allocate the input buffer region with a 256-byte alignment and a size of a multiple of 256-byte.
- Allocate the maximum access size (the maximum size (*maxEsSize*) of elementary stream) from *pEs* of `SceAudiodecCtrl`, which is the input starting address, within the input buffer. However, there are no alignment restrictions for *pEs*.
- Do not specify the same input buffer region for multiple decoders at the same time.

Figure 1 Restrictions on the input buffer

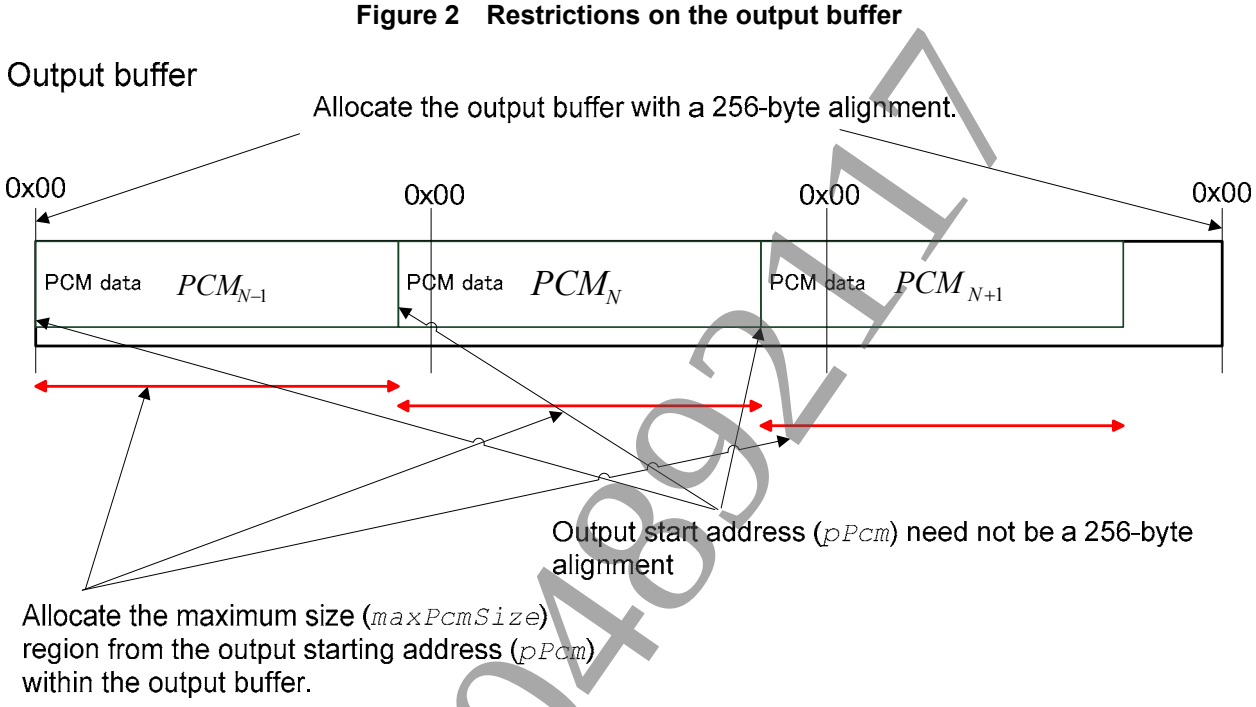


Output Buffer

PCM data is written to the output buffer region of `sceAudiodecDecode()` in the Codec Engine. To maintain cache coherency on the ARM and Codec Engine sides at this time, the following restrictions apply. Also refer to Figure 2.

- Allocate the output buffer region with a 256-byte alignment and a size of a multiple of 256-byte.
- Allocate the maximum size (*maxPcmSize*) of the PCM that is to be output from *pPcm* of `SceAudiodecCtrl`, which is the output starting address, within the output buffer. However, there are no alignment restrictions for *pPcm*.

Figure 2 Restrictions on the output buffer



Output PCM Data

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

Memory Allocated to the Audio Decoder

Normally, system memory is allocated to the memory of the audio decoder. The APIs used at this time are `sceAudiodecInitLibrary()`, `sceAudiodecTermLibrary()`, `sceAudiodecCreateDecoder()`, and `sceAudiodecDeleteDecoder()`.

The user memory can be specified and this can be allocated to the audio decoder. The APIs used at this time are `sceAudiodecCreateDecoderExternal()` and `sceAudiodecDeleteDecoderExternal()`. `sceAudiodecInitLibrary()` and `sceAudiodecTermLibrary()` do not need to be used.

Decoding is performed using `sceAudiodecDecode()`, the common decoding API for audio decoders generated using any of these methods. In addition, audio decoders generated using any of these methods can co-exist.

However, currently, ATRAC9™ and AAC are the only audio decoders that can specify the user memory.

ATRAC9™ Band Extension Support

libaudiodec does not support the ATRAC9™ Band Extension, which is an extended format of ATRAC9™.

Check the value of `dwVersionInfo` in the RIFF Header to see whether the format is ATRAC9™ Band Extension. For details, refer to the "ATRAC9™ File Format" document.