**EncoderDecoder Module Description Document**

**1.Introduction**

The audio encoder mainly compresses PCM data into an audio bitstream using specific algorithms to reduce storage space or transmission bandwidth. The currently supported audio encoding formats include: ADPCM, SBC, and MP2/MP3.

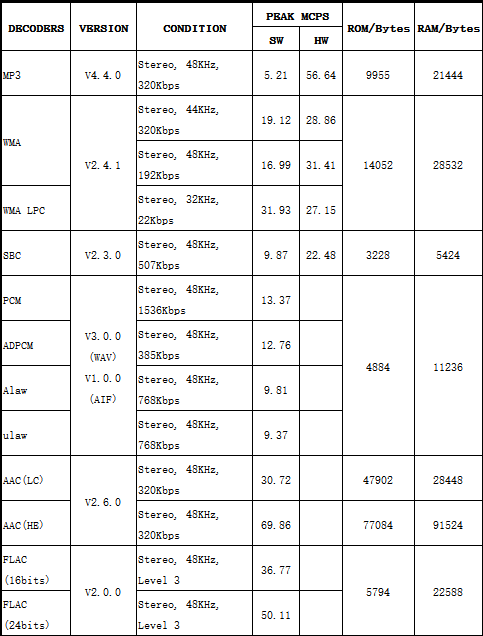
The audio decoder mainly decompresses audio files or audio bitstreams into PCM data using specific algorithms. The currently supported audio decoding formats are shown in the table below.

Table 1. List of Supported Audio Decoder Formats

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Format | Extension Name | Sampling Rate | Vocal Channel | Bit Rate |
| MP3 | .mp2, .mp3 | 8/11.025/12/16/22.05/24/32/44.1/48KHz | 1 or 2 | 32kbps~384kbps |
| WMA(standard) | .wma, .asf, .wmv | 8/11.025/16/22.05/32/44.1/48KHz | 1 or 2 | 0kbps~320kbps |
| PCM(raw) | .wav, .aif,  .aiff | 8/11.025/12/16/22.05/24/32/44.1/48KHz | 1 or 2 | 64kbps~3072kbps |
| PCM(u-law, A-law) | .wav, .aif,  .aifc | 8/11.025/12/16/22.05/24/32/44.1/48KHz | 1 or 2 | 64kbps~768kbps |
| IMA-ADPCM | .wav, .aif,  .aifc | 8/11.025/12/16/22.05/24/32/44.1/48KHz | 1 or 2 | 32kbps~384kbps |
| AAC-LC | .m4a, .aac,  .mp4, .mov | 8/11.025/12/16/22.05/24/32/44.1/48KHz | 1 or 2 | all bitrates  allowed |
| HE-AAC v1 (eAAC or AAC+) | .m4a, .aac,  .mp4, .mov | 8/11.025/12/16/22.05/24/32/44.1/48KHz | 1 or 2 | all bitrates  allowed |
| HE-AAC v2 (eAAC+or AAC++) | .m4a, .aac,  .mp4, .mov | 8/11.025/12/16/22.05/24/32/44.1/48KHz | 1 or 2 | all bitrates  allowed |
| FLAC | .flac | 8/16/22.05/24/32/44.1/48KHz | 1 or 2 | all bitrates  allowed |

Among them, WAV, FALC, AAC, and AIF are pure software decoders, while MP3, WMA, and SBC are combined software and hardware decoders. The performance indicators of each decoder are shown in the table below:

Table 2. Audio Decoder Performance Statistics



**2.Main Features**

* Supports ADPCM, SBC, MP2/MP3 encoding.
* Supports ADPCM, A-law, u-law, SBC, MP2/MP3, WMA, AAC, and FLAC decoding.

**3.Instructions for Use**

**3.1. Encoder Usage**

1. Initialize the encoder. Continue with the next step after successful initialization.

2. If the PCM data to be encoded is greater than or equal to one frame of data, call the encoder to perform encoding. After successful encoding, the encoded data will be output.

3. Repeat step 2.

21 #include "audio\_encoder.h"

 22

 23 **void** AudioEncoderProcess**(void)**

 24 {

 25     **if**(RT\_SUCCESS == audio\_encoder\_initialize(...)

 26     {

 27         **while**(1)

 28         {

 29             **if**(pcm\_samples\_avariable() >= SAMPLES\_PER\_FRAME)

 30             {

 31                 **if**(RT\_SUCCESS == audio\_encoder\_encode(...))

 32                 {

 33                     //Get the encoded data to save or transfer

 34                     //TODO

 35                 }

 36             }

 37         }

 38     }

 39 }

**3.2. Decoder Usage**

1. Initialize the decoder. Continue with the next step after successful initialization.
2. Check whether the decoder can continue decoding. If yes, proceed with the following steps; if not, exit the decoder.
3. Call the frame decoding API function. A return value of RT\_SUCCESS indicates that software decoding was successful.
4. Check whether the decoder includes hardware decoding. If yes, wait for the hardware decoding to complete.
5. After hardware decoding is completed, one frame of PCM data will be obtained, which can be used for playback or processing.

42 #include "audio\_decoder.h"

43

44 **void** AudioDecoderProcess(**void**)

45 {

46    **if**(RT\_SUCCESS == audio\_decoder\_initialize((**uint8\_t**\*)VMEM\_ADDR, fp, IO\_TYPE\_FILE, MP3\_DECODER))

47     {

48        **if**(RT\_YES == audio\_decoder\_can\_continue())

49         {

 50             **if**(RT\_SUCCESS == audio\_decoder\_decode())

 51             {

 52                 **if**(is\_audio\_decoder\_with\_hardware())

 53                 {

 54                     audio\_decoder\_wait\_for\_hardware\_decoder\_done();

 55                 }

 56

 57                 //Get PCM data

 58                 //TODO

 59             }

 60         }

 61     }

 62 }

Figure 2. Pseudocode of the Audio Decoder Decoding Process