Cyberon Speex Programming Guide

Version: 1.0

Date of issue: 2020/12/22

Cyberon Corporation

Software solution provider for embedded system

http://www.cyberon.com.tw/

Contents

History：

Ver.	Data	Update
1.0	2020-12-22	1. First release.

1. Modified Speex

1.1. Introduction

Speex is a free open source audio codec (https://www.speex.org/). Cyberon modify it for support only parts of its feature:

1. It compress audio data with 8 or 16 KHz, mono channel, 16 bits, PCM format. We remove the support of 32 KHz sampling rate or stereo channel audio.
2. For 8 KHz audio, the supported compression bit rate is 4000, 6000, 8000, 11200 and 15200 bps. 8000 bits per second is recommended.
3. For 16 KHz audio, the supported compression bit rate is 8000, 10000, 12800, 16800, 20800 and 24000 bps. 12800 or 16800 bits per second is recommended.
4. We only support constant bit rate(don't support variable bit rate).
5. Simplify and redesign the encode/decode API.

1.2. Specification

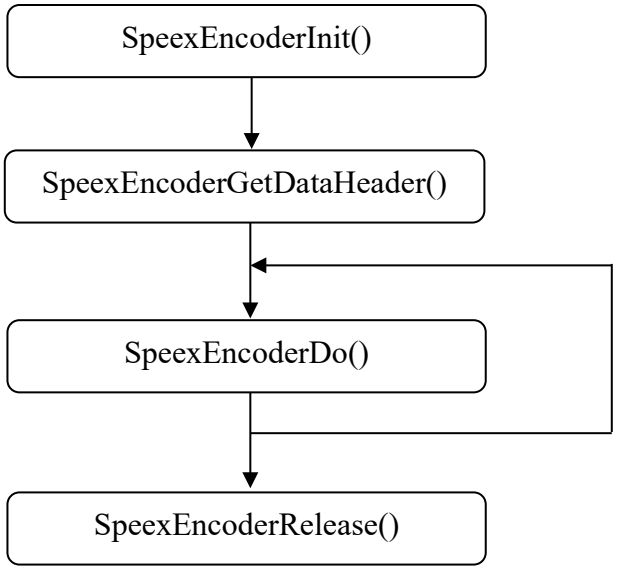
1. Code & table: 65 K
2. RAM: 3 KB heap and 6 KB stack per decode or encode stream.
3. The CPU usage is depend on sampling rate and bit rate, we list three case:

Sampling rate	8 KHz	16 KHz	
Bit rate	8000 bps	12800 bps	16800 bps
Encode	42.5 MCPS	72.8 MCPS	60 MCPS
Decode	5.9 MCPS	13 MCPS	13 MCPS

PS. Test on RA6M1 CM4 120 MHz.CPU with -O3 compile option.

2. Encode SDK

2.1. Calling Flow Chart



2.2. API Function

SpeexEncoderInit

Initialize a speex encoder object.

Prototype

HANDLE SpeexEncoderInit(int nChannelNum, int nSampleRate, int nExpectBitRate, int nComlexity, int *pnErr)

Parameters

nChannelNum (IN):

At current time, we only support mono channel, so it must be 1.

nSampleRate (IN):

It could be 8000 or 16000.

nExpectBitRate (IN):

The expect bit rate for encoder. The valid range is 2400 ~ 64000.

This value should be a multiple of 400. The real bit rate will near to this value and can be got in SpeexDataHeader.nBitRate. We use constant bit rate to encode.

For 8 KHz narrow-band, the supported SpeexDataHeader.nBitRate are:

4000(very poor), 6000(poor), 8000(acceptable), 11200(acceptable), 15200(acceptable).
For 16 KHz wide-band, the supported SpeexDataHeader.nBitRate are:
8000(poor), 10000(bad), 12800(good), 16800(better), 20800(good enough), 24000(very good).

nComlexity (IN):

The valid range is 1 ~ 10. It require more CPU power for the larger value.
When nComlexity is 10, become very slow but get better quality.

pnErr (OUT):

The error code.

Return value

The handle of speex encoder object or NULL if error.

SpeexEncoderRelease

Release a speex encoder object.

Prototype

int SpeexEncoderRelease(HANDLE hSpeexEncoder)

Parameters

hSpeexEncoder (IN):

The handle of speex encoder object.

Return value

SPEEX_API_SUCCESS(0) if success, negative value if error.

SpeexEncoderGetDataHeader

Get the data header of speex.

Prototype

int SpeexEncoderGetDataHeader(HANDLE hSpeexEncoder, SpeexDataHeader *lpoSpeexDataHeader)

Parameters

hSpeexEncoder (IN):

The handle of speex encoder object.

lpoSpeexDataHeader (OUT):

The structure of speex data header, it contains many properties of speex encode data.
We put it to the header of *.spx file. The speex decoder need these information to

do initialization.

Return value

SPEEX_API_SUCCESS(0) if success, negative value if error.

SpeexEncoderDo

Encode the input PCM frame to output data.

Prototype

int SpeexEncoderDo(HANDLE hSpeexEncoder, const short *lpsInFrame, int nInFrameSamples, BYTE *lpbyOutEncode, int nOutEncodeSize)

Parameters

hSpeexEncoder (IN):

The handle of speex encoder object.

lpsInFrame (IN):

The input 16 bits, mono PCM data frame.

nInFrameSamples (IN):

The sample count of lpsInFrame, it must be equal to SpeexDataHeader.nFrameSamples. The duration of a speex frame is 20 ms. So it is 160 for 8KHz, 320 for 16KHz.

lpbyOutEncode (OUT):

The output of encoded data.

nOutEncodeSize (IN):

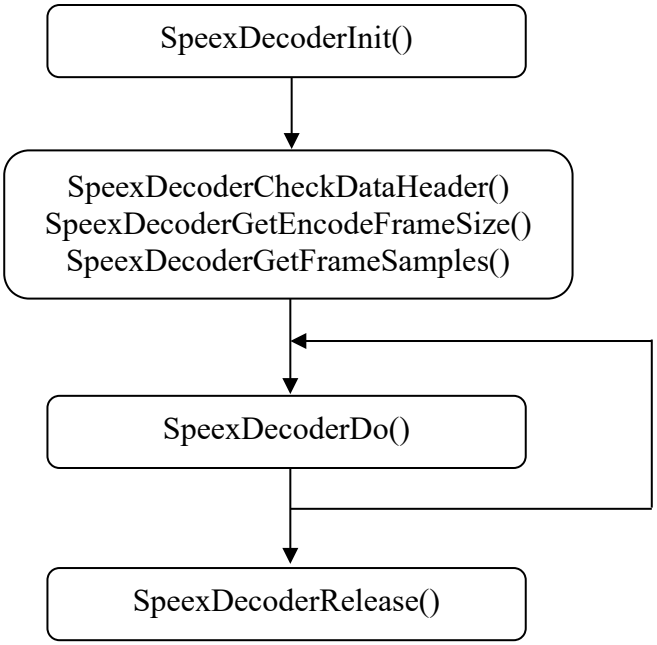
The byte count of lpbyOutEncode, it must be equal to SpeexDataHeader.nEncodeSize. There are 50 frames in one second, so it is equal to SpeexDataHeader.nBitRate/8/50.

Return value

SPEEX_API_SUCCESS(0) if success, negative value if error.

3. Decode SDK

3.1. Calling Flow Chart



3.2. API Function

SpeexDecoderInit

Initialize a speex decoder object.

Prototype

HANDLE SpeexDecoderInit(int nChannelNum, int nSampleRate, int nBitRate, BOOL bUseEnhancement, int *pnErr)

Parameters

nChannelNum (IN):

At current time, we only support mono channel, so it must be 1.

nSampleRate (IN):

It could be 8000 or 16000.

nBitRate (IN):

The data bit rate. Please refer to SpeexDataHeader.nBitRate.

bUseEnhancement (IN):

We use it for a little better quality but more CPU power.

pnErr (OUT):

The error code.

Return value

The handle of speex decoder object or NULL if error.

SpeexDecoderRelease

Release a speex decoder object.

Prototype

int SpeexDecoderRelease(HANDLE hSpeexDecoder)

Parameters

hSpeexDecoder (IN):

The handle of speex decoder object.

Return value

SPEEX_API_SUCCESS(0) if success, negative value if error.

SpeexDecoderCheckDataHeader

Check the validation of speex data header.

Prototype

int SpeexDecoderCheckDataHeader(const SpeexDataHeader *lpoSpeexDataHeader)

Parameters

lpoSpeexDataHeader (IN):

The speex data header.

Return value

SPEEX_API_SUCCESS(0) if success, negative value if error.

SpeexDecoderGetEncodeFrameSize

Get the encode size per frame.

Prototype

int SpeexDecoderGetEncodeFrameSize(HANDLE hSpeexDecoder)

Parameters

hSpeexDecoder (IN):

The handle of speex decoder object.

Return value

The encode size per frame, negative value if error.

SpeexDecoderGetFrameSamples

Get the PCM sample count per frame.

Prototype

int SpeexDecoderGetEncodeFrameSize(HANDLE hSpeexDecoder)

Parameters

hSpeexDecoder (IN):

The handle of speex decoder object.

Return value

The PCM sample count per frame, negative value if error.

SpeexDecoderDo

Decode the input data to PCM samples.

Prototype

int SpeexDecoderDo(HANDLE hSpeexDecoder, const BYTE *lpbyEncode, int nEncodeSize, short *lpsOutFrame, int nOutFrameSamples)

Parameters

hSpeexDecoder (IN):

The handle of speex decoder object.

lpbyEncode (IN):

The input of encoded data.

nEncodeSize (IN):

The byte count of lpbyEncode, it must be equal to the return value of SpeexDecoderGetEncodeSize().

lpsOutFrame (OUT):

The output 16 bits PCM data frame.

nOutEncodeSize (IN):

The sample count of lpsOutFrame, it must be equal to the return value of SpeexDecoderGetFrameSamples().

Return value

SPEEX_API_SUCCESS(0) if success, negative value if error.