Speex Reference Manual

1.1.12

Generated by Doxygen 1.4.6

Sun Aug 6 13:29:22 2006

Contents

Chapter 1

Speex Directory Hierarchy

1.1 Speex Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 2

Speex Hierarchical Index

2.1 Speex Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Speex Class Index

3.1 Speex Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Speex File Index

4.1 Speex File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Speex Directory Documentation

5.1 include/ Directory Reference

Directories

- directory speex

5.2 include/speex/ Directory Reference

Files

- file speex.h

  Describes the different modes of the codec.

- file speex_bits.h

  Handles bit packing/unpacking.

- file speex_callbacks.h

  Describes callback handling and in-band signalling.

- file speex_echo.h

  Echo cancellation.

- file speex_header.h

  Describes the Speex header.

- file speex_jitter.h

  Adaptive jitter buffer for Speex.

- file speex_noglobals.h

  Dynamically allocates the different modes of the codec.

- file speex_preprocess.h

  Speex preprocessor.

- file speex_stereo.h

  Describes the handling for intensity stereo.

- file speex_types.h

  Speex types.

Chapter 6

Speex Class Documentation

6.1 SpeexBits Struct Reference

`#include <speex_bits.h>`

Public Attributes

- char * chars
- int nbBits
- int charPtr
- int bitPtr
- int owner
- int overflow
- int buf_size
- int reserved1
- void * reserved2

6.1.1 Detailed Description

Bit-packing data structure representing (part of) a bit-stream.

6.1.2 Member Data Documentation

6.1.2.1 int SpeexBits::bitPtr

Position of the bit "cursor" within the current char

6.1.2.2 int SpeexBits::buf_size

Allocated size for buffer

6.1.2.3 int SpeexBits::charPtr

Position of the byte "cursor"

6.1.2.4 char* SpeexBits::chars

"raw" data

6.1.2.5 int SpeexBits::nbBits

Total number of bits stored in the stream

6.1.2.6 int SpeexBits::overflow

Set to one if we try to read past the valid data

6.1.2.7 int SpeexBits::owner

Does the struct "own" the "raw" buffer (member "chars")

6.1.2.8 int SpeexBits::reserved1

Reserved for future use

6.1.2.9 void* SpeexBits::reserved2

Reserved for future use

The documentation for this struct was generated from the following file:

- speex_bits.h

6.2 SpeexCallback Struct Reference

`#include <speex_callbacks.h>`

Public Attributes

- int callback_id
- speex_callback_func func
- void * data
- void * reserved1
- int reserved2

6.2.1 Detailed Description

Callback information

6.2.2 Member Data Documentation

6.2.2.1 int SpeexCallback::callback_id

ID associated to the callback

6.2.2.2 void* SpeexCallback::data

Data that will be sent to the handler

6.2.2.3 speex_callback_func SpeexCallback::func

Callback handler function

6.2.2.4 void* SpeexCallback::reserved1

Reserved for future use

6.2.2.5 int SpeexCallback::reserved2

Reserved for future use

The documentation for this struct was generated from the following file:

- speex_callbacks.h

6.3 SpeexHeader Struct Reference

`#include <speex_header.h>`

Public Attributes

- char speex_string [SPEEX_HEADER_STRING_LENGTH]
- char speex_version [SPEEX_HEADER_VERSION_LENGTH]
- spx_int32_t speex_version_id
- spx_int32_t header_size
- spx_int32_t rate
- spx_int32_t mode
- spx_int32_t mode_bitstream_version
- spx_int32_t nb_channels
- spx_int32_t bitrate
- spx_int32_t frame_size
- spx_int32_t vbr
- spx_int32_t frames_per_packet
- spx_int32_t extra_headers
- spx_int32_t reserved1
- spx_int32_t reserved2

6.3.1 Detailed Description

Speex header info for file-based formats

6.3.2 Member Data Documentation

6.3.2.1 spx_int32_t SpeexHeader::bitrate

Bit-rate used

6.3.2.2 spx_int32_t SpeexHeader::extra_headers

Number of additional headers after the comments

6.3.2.3 spx_int32_t SpeexHeader::frame_size

Size of frames

6.3.2.4 spx_int32_t SpeexHeader::frames_per_packet

Number of frames stored per Ogg packet

6.3.2.5 spx_int32_t SpeexHeader::header_size

Total size of the header (sizeof(SpeexHeader))

6.3.2.6 spx_int32_t SpeexHeader::mode

Mode used (0 for narrowband, 1 for wideband)

6.3.2.7 spx_int32_t SpeexHeader::mode_bitstream_version

Version ID of the bit-stream

6.3.2.8 spx_int32_t SpeexHeader::nb_channels

Number of channels encoded

6.3.2.9 spx_int32_t SpeexHeader::rate

Sampling rate used

6.3.2.10 spx_int32_t SpeexHeader::reserved1

Reserved for future use, must be zero

6.3.2.11 spx_int32_t SpeexHeader::reserved2

Reserved for future use, must be zero

6.3.2.12 char SpeexHeader::speex_string[SPEEX_HEADER_STRING_LENGTH]

Identifies a Speex bit-stream, always set to "Speex "

6.3.2.13 char SpeexHeader::speex_version[SPEEX_HEADER_VERSION_LENGTH]

Speex version

6.3.2.14 spx_int32_t SpeexHeader::speex_version_id

Version for Speex (for checking compatibility)

6.3.2.15 spx_int32_t SpeexHeader::vbr

1 for a VBR encoding, 0 otherwise

The documentation for this struct was generated from the following file:

- speex_header.h

6.4 SpeexJitter Struct Reference

`#include <speex_jitter.h>`

Public Attributes

- SpeexBits current_packet
- int valid_bits
- JitterBuffer * **packets**
- void * dec
- int frame_size

6.4.1 Detailed Description

Speex jitter-buffer state.

6.4.2 Member Data Documentation

6.4.2.1 SpeexBits SpeexJitter::current_packet

Current Speex packet

6.4.2.2 void* SpeexJitter::dec

Pointer to Speex decoder

6.4.2.3 int SpeexJitter::frame_size

Frame size of Speex decoder

6.4.2.4 int SpeexJitter::valid_bits

True if Speex bits are valid

The documentation for this struct was generated from the following file:

- speex_jitter.h

6.5 SpeexMode Struct Reference

`#include <speex.h>`

Public Attributes

- const void * mode
- mode_query_func query
- const char * modeName
- int modeID
- int bitstream_version
- encoder_init_func enc_init
- encoder_destroy_func enc_destroy
- encode_func enc
- decoder_init_func dec_init
- decoder_destroy_func dec_destroy
- decode_func dec
- encoder_ctl_func enc_ctl
- decoder_ctl_func dec_ctl

6.5.1 Detailed Description

Struct defining a Speex mode

6.5.2 Member Data Documentation

6.5.2.1 int SpeexMode::bitstream_version

Version number of the bitstream (incremented every time we break bitstream compatibility

6.5.2.2 decode_func SpeexMode::dec

Pointer to frame decoding function

6.5.2.3 decoder_ctl_func SpeexMode::dec_ctl

ioctl-like requests for decoder

6.5.2.4 decoder_destroy_func SpeexMode::dec_destroy

Pointer to decoder destruction function

6.5.2.5 decoder_init_func SpeexMode::dec_init

Pointer to decoder initialization function

6.5.2.6 encode_func SpeexMode::enc

Pointer to frame encoding function

6.5.2.7 encoder_ctl_func SpeexMode::enc_ctl

ioctl-like requests for encoder

6.5.2.8 encoder_destroy_func SpeexMode::enc_destroy

Pointer to encoder destruction function

6.5.2.9 encoder_init_func SpeexMode::enc_init

Pointer to encoder initialization function

6.5.2.10 const void* SpeexMode::mode

Pointer to the low-level mode data

6.5.2.11 int SpeexMode::modeID

ID of the mode

6.5.2.12 const char* SpeexMode::modeName

The name of the mode (you should not rely on this to identify the mode)

6.5.2.13 mode_query_func SpeexMode::query

Pointer to the mode query function

The documentation for this struct was generated from the following file:

- speex.h

6.6 SpeexPreprocessState Struct Reference

`#include <speex_preprocess.h>`

Public Attributes

- int frame_size
- int ps_size
- int sampling_rate
- int **denoise_enabled**
- int **agc_enabled**
- float **agc_level**
- int **vad_enabled**
- int **dereverb_enabled**
- float **reverb_decay**
- float **reverb_level**
- float **speech_prob_start**
- float **speech_prob_continue**
- float * frame
- float * ps
- float * gain2
- float * window
- float * noise
- float * reverb_estimate
- float * old_ps
- float * gain
- float * prior
- float * post
- float * S
- float * Smin
- float * Stmp
- float * update_prob
- float * zeta
- float **Zpeak**
- float **Zlast**
- float * loudness_weight
- float * **echo_noise**
- float * **noise_bands**
- float * **noise_bands2**
- int **noise_bandsN**
- float * **speech_bands**
- float * **speech_bands2**
- int **speech_bandsN**
- float * inbuf
- float * outbuf
- float **speech_prob**
- int **last_speech**
- float loudness
- float loudness2

- int nb_adapt
- int nb_loudness_adapt
- int consec_noise
- int nb_preprocess
- drft_lookup * fft_lookup

6.6.1 Detailed Description

Speex pre-processor state.

6.6.2 Member Data Documentation

6.6.2.1 int SpeexPreprocessState::consec_noise

Number of consecutive noise frames

6.6.2.2 struct drft_lookup* SpeexPreprocessState::fft_lookup

Lookup table for the FFT

6.6.2.3 float* SpeexPreprocessState::frame

Processing frame (2*ps_size)

6.6.2.4 int SpeexPreprocessState::frame_size

Number of samples processed each time

6.6.2.5 float* SpeexPreprocessState::gain

Ephraim Malah gain

6.6.2.6 float* SpeexPreprocessState::gain2

Adjusted gains

6.6.2.7 float* SpeexPreprocessState::inbuf

Input buffer (overlapped analysis)

6.6.2.8 float SpeexPreprocessState::loudness

loudness estimate

6.6.2.9 float SpeexPreprocessState::loudness2

loudness estimate

6.6.2.10 float* SpeexPreprocessState::loudness_weight

Perceptual loudness curve

6.6.2.11 int SpeexPreprocessState::nb_adapt

Number of frames used for adaptation so far

6.6.2.12 int SpeexPreprocessState::nb_loudness_adapt

Number of frames used for loudness adaptation so far

6.6.2.13 int SpeexPreprocessState::nb_preprocess

Number of frames processed so far

6.6.2.14 float* SpeexPreprocessState::noise

Noise estimate

6.6.2.15 float* SpeexPreprocessState::old_ps

Power spectrum for last frame

6.6.2.16 float* SpeexPreprocessState::outbuf

Output buffer (for overlap and add)

6.6.2.17 float* SpeexPreprocessState::post

A-posteriori SNR

6.6.2.18 float* SpeexPreprocessState::prior

A-priori SNR

6.6.2.19 float* SpeexPreprocessState::ps

Current power spectrum

6.6.2.20 int SpeexPreprocessState::ps_size

Number of points in the power spectrum

6.6.2.21 float* SpeexPreprocessState::reverb_estimate

Estimate of reverb energy

6.6.2.22 float* SpeexPreprocessState::S

Smoothed power spectrum

6.6.2.23 int SpeexPreprocessState::sampling_rate

Sampling rate of the input/output

6.6.2.24 float* SpeexPreprocessState::Smin

See Cohen paper

6.6.2.25 float* SpeexPreprocessState::Stmp

See Cohen paper

6.6.2.26 float* SpeexPreprocessState::update_prob

Propability of speech presence for noise update

6.6.2.27 float* SpeexPreprocessState::window

Analysis/Synthesis window

6.6.2.28 float* SpeexPreprocessState::zeta

Smoothed a priori SNR

The documentation for this struct was generated from the following file:

- speex_preprocess.h

6.7 SpeexStereoState Struct Reference

`#include <speex_stereo.h>`

Public Attributes

- float balance
- float e_ratio
- float smooth_left
- float smooth_right
- float reserved1
- float reserved2

6.7.1 Detailed Description

State used for decoding (intensity) stereo information

6.7.2 Member Data Documentation

6.7.2.1 float SpeexStereoState::balance

Left/right balance info

6.7.2.2 float SpeexStereoState::e_ratio

Ratio of energies: E(left+right)/[E(left)+E(right)]

6.7.2.3 float SpeexStereoState::reserved1

Reserved for future use

6.7.2.4 float SpeexStereoState::reserved2

Reserved for future use

6.7.2.5 float SpeexStereoState::smooth_left

Smoothed left channel gain

6.7.2.6 float SpeexStereoState::smooth_right

Smoothed right channel gain

The documentation for this struct was generated from the following file:

- speex_stereo.h

Chapter 7

Speex File Documentation

7.1 speex.h File Reference

Describes the different modes of the codec.

```
#include "speex/speex_bits.h"
#include "speex/speex_types.h"
```

Classes

- struct SpeexMode

Defines

- #define SPEEX_SET_ENH 0
- #define SPEEX_GET_ENH 1
- #define SPEEX_GET_FRAME_SIZE 3
- #define SPEEX_SET_QUALITY 4
- #define SPEEX_SET_MODE 6
- #define SPEEX_GET_MODE 7
- #define SPEEX_SET_LOW_MODE 8
- #define SPEEX_GET_LOW_MODE 9
- #define SPEEX_SET_HIGH_MODE 10
- #define SPEEX_GET_HIGH_MODE 11
- #define SPEEX_SET_VBR 12
- #define SPEEX_GET_VBR 13
- #define SPEEX_SET_VBR_QUALITY 14
- #define SPEEX_GET_VBR_QUALITY 15
- #define SPEEX_SET_COMPLEXITY 16
- #define SPEEX_GET_COMPLEXITY 17
- #define SPEEX_SET_BITRATE 18
- #define SPEEX_GET_BITRATE 19
- #define SPEEX_SET_HANDLER 20
- #define SPEEX_SET_USER_HANDLER 22
- #define SPEEX_SET_SAMPLING_RATE 24

- #define SPEEX_GET_SAMPLING_RATE 25
- #define SPEEX_RESET_STATE 26
- #define SPEEX_GET_RELATIVE_QUALITY 29
- #define SPEEX_SET_VAD 30
- #define SPEEX_GET_VAD 31
- #define SPEEX_SET_ABR 32
- #define SPEEX_GET_ABR 33
- #define SPEEX_SET_DTX 34
- #define SPEEX_GET_DTX 35
- #define SPEEX_SET_SUBMODE_ENCODING 36
- #define SPEEX_GET_SUBMODE_ENCODING 37
- #define SPEEX_GET_LOOKAHEAD 39
- #define SPEEX_SET_PLC_TUNING 40
- #define SPEEX_GET_PLC_TUNING 41
- #define SPEEX_SET_VBR_MAX_BITRATE 42
- #define SPEEX_GET_VBR_MAX_BITRATE 43
- #define SPEEX_SET_HIGHPASS 44
- #define SPEEX_GET_HIGHPASS 45
- #define SPEEX_GET_PI_GAIN 100
- #define SPEEX_GET_EXC 101
- #define SPEEX_GET_INNOV 102
- #define SPEEX_GET_DTX_STATUS 103
- #define SPEEX_SET_INNOVATION_SAVE 104
- #define SPEEX_SET_WIDEBAND 105
- #define SPEEX_SET_PF 0
- #define SPEEX_GET_PF 1
- #define SPEEX_MODE_FRAME_SIZE 0
- #define SPEEX_SUBMODE_BITS_PER_FRAME 1
- #define SPEEX_LIB_GET_MAJOR_VERSION 1
- #define SPEEX_LIB_GET_MINOR_VERSION 3
- #define SPEEX_LIB_GET_MICRO_VERSION 5
- #define SPEEX_LIB_GET_EXTRA_VERSION 7
- #define SPEEX_LIB_GET_VERSION_STRING 9
- #define SPEEX_NB_MODES 3
- #define SPEEX_MODEID_NB 0
- #define SPEEX_MODEID_WB 1
- #define SPEEX_MODEID_UWB 2

Typedefs

- typedef void *(* encoder_init_func)(const struct SpeexMode *mode)
- typedef void(* encoder_destroy_func)(void *st)
- typedef int(* encode_func)(void *state, void *in, SpeexBits *bits)
- typedef int(* encoder_ctl_func)(void *state, int request, void *ptr)
- typedef void *(* decoder_init_func)(const struct SpeexMode *mode)
- typedef void(* decoder_destroy_func)(void *st)
- typedef int(* decode_func)(void *state, SpeexBits *bits, void *out)
- typedef int(* decoder_ctl_func)(void *state, int request, void *ptr)
- typedef int(* mode_query_func)(const void *mode, int request, void *ptr)

Functions

- void * speex_encoder_init (const SpeexMode *mode)
- void speex_encoder_destroy (void *state)
- int speex_encode (void *state, float *in, SpeexBits *bits)
- int speex_encode_int (void *state, spx_int16_t *in, SpeexBits *bits)
- int speex_encoder_ctl (void *state, int request, void *ptr)
- void * speex_decoder_init (const SpeexMode *mode)
- void speex_decoder_destroy (void *state)
- int speex_decode (void *state, SpeexBits *bits, float *out)
- int speex_decode_int (void *state, SpeexBits *bits, spx_int16_t *out)
- int speex_decoder_ctl (void *state, int request, void *ptr)
- int speex_mode_query (const SpeexMode *mode, int request, void *ptr)
- int speex_lib_ctl (int request, void *ptr)
- const SpeexMode * speex_lib_get_mode (int mode)

Variables

- const SpeexMode speex_nb_mode
- const SpeexMode speex_wb_mode
- const SpeexMode speex_uwb_mode
- const SpeexMode *const speex_mode_list [SPEEX_NB_MODES]

7.1.1 Detailed Description

Describes the different modes of the codec.

7.1.2 Define Documentation

7.1.2.1 #define SPEEX_GET_ABR 33

Get Average Bit-Rate (ABR) setting (in bps)

7.1.2.2 #define SPEEX_GET_BITRATE 19

Get current bit-rate used by the encoder or decoder

7.1.2.3 #define SPEEX_GET_COMPLEXITY 17

Get current complexity of the encoder (0-10)

7.1.2.4 #define SPEEX_GET_DTX 35

Get DTX status (1 for on, 0 for off)

7.1.2.5 #define SPEEX_GET_DTX_STATUS 103

Used internally

7.1.2.6 #define SPEEX_GET_ENH 1

Get enhancement state (decoder only)

7.1.2.7 #define SPEEX_GET_EXC 101

Used internally

7.1.2.8 #define SPEEX_GET_FRAME_SIZE 3

Obtain frame size used by encoder/decoder

7.1.2.9 #define SPEEX_GET_HIGH_MODE 11

Get current high-band mode in use (wideband only)

7.1.2.10 #define SPEEX_GET_HIGHPASS 45

Get status of input/output high-pass filtering

7.1.2.11 #define SPEEX_GET_INNOV 102

Used internally

7.1.2.12 #define SPEEX_GET_LOOKAHEAD 39

Returns the lookahead used by Speex

7.1.2.13 #define SPEEX_GET_LOW_MODE 9

Get current low-band mode in use (wideband only)

7.1.2.14 #define SPEEX_GET_MODE 7

Get current sub-mode in use

7.1.2.15 #define SPEEX_GET_PF 1

Equivalent to SPEEX_GET_ENH

7.1.2.16 #define SPEEX_GET_PI_GAIN 100

Used internally

7.1.2.17 #define SPEEX_GET_PLC_TUNING 41

Gets tuning for PLC

7.1.2.18 #define SPEEX_GET_RELATIVE_QUALITY 29

Get VBR info (mostly used internally)

7.1.2.19 #define SPEEX_GET_SAMPLING_RATE 25

Get sampling rate used in bit-rate computation

7.1.2.20 #define SPEEX_GET_SUBMODE_ENCODING 37

Get submode encoding in each frame

7.1.2.21 #define SPEEX_GET_VAD 31

Get VAD status (1 for on, 0 for off)

7.1.2.22 #define SPEEX_GET_VBR 13

Get VBR status (1 for on, 0 for off)

7.1.2.23 #define SPEEX_GET_VBR_MAX_BITRATE 43

Gets the max bit-rate allowed in VBR mode

7.1.2.24 #define SPEEX_GET_VBR_QUALITY 15

Get current quality value for VBR encoding (0-10)

7.1.2.25 #define SPEEX_LIB_GET_EXTRA_VERSION 7

Get extra Speex version

7.1.2.26 #define SPEEX_LIB_GET_MAJOR_VERSION 1

Get major Speex version

7.1.2.27 #define SPEEX_LIB_GET_MICRO_VERSION 5

Get micro Speex version

7.1.2.28 #define SPEEX_LIB_GET_MINOR_VERSION 3

Get minor Speex version

7.1.2.29 #define SPEEX_LIB_GET_VERSION_STRING 9

Get Speex version string

7.1.2.30 #define SPEEX_MODE_FRAME_SIZE 0

Query the frame size of a mode

7.1.2.31 #define SPEEX_MODEID_NB 0

modeID for the defined narrowband mode

7.1.2.32 #define SPEEX_MODEID_UWB 2

modeID for the defined ultra-wideband mode

7.1.2.33 #define SPEEX_MODEID_WB 1

modeID for the defined wideband mode

7.1.2.34 #define SPEEX_NB_MODES 3

Number of defined modes in Speex

7.1.2.35 #define SPEEX_RESET_STATE 26

Reset the encoder/decoder memories to zero

7.1.2.36 #define SPEEX_SET_ABR 32

Set Average Bit-Rate (ABR) to n bits per seconds

7.1.2.37 #define SPEEX_SET_BITRATE 18

Set bit-rate used by the encoder (or lower)

7.1.2.38 #define SPEEX_SET_COMPLEXITY 16

Set complexity of the encoder (0-10)

7.1.2.39 #define SPEEX_SET_DTX 34

Set DTX status (1 for on, 0 for off)

7.1.2.40 #define SPEEX_SET_ENH 0

Set enhancement on/off (decoder only)

7.1.2.41 #define SPEEX_SET_HANDLER 20

Define a handler function for in-band Speex request

7.1.2.42 #define SPEEX_SET_HIGH_MODE 10

Set high-band sub-mode to use (wideband only)

7.1.2.43 #define SPEEX_SET_HIGHPASS 44

Turn on/off input/output high-pass filtering

7.1.2.44 #define SPEEX_SET_INNOVATION_SAVE 104

Used internally

7.1.2.45 #define SPEEX_SET_LOW_MODE 8

Set low-band sub-mode to use (wideband only)

7.1.2.46 #define SPEEX_SET_MODE 6

Set sub-mode to use

7.1.2.47 #define SPEEX_SET_PF 0

Equivalent to SPEEX_SET_ENH

7.1.2.48 #define SPEEX_SET_PLC_TUNING 40

Sets tuning for packet-loss concealment (expected loss rate)

7.1.2.49 #define SPEEX_SET_QUALITY 4

Set quality value

7.1.2.50 #define SPEEX_SET_SAMPLING_RATE 24

Set sampling rate used in bit-rate computation

7.1.2.51 #define SPEEX_SET_SUBMODE_ENCODING 36

Set submode encoding in each frame (1 for yes, 0 for no, setting to no breaks the standard)

7.1.2.52 #define SPEEX_SET_USER_HANDLER 22

Define a handler function for in-band user-defined request

7.1.2.53 #define SPEEX_SET_VAD 30

Set VAD status (1 for on, 0 for off)

7.1.2.54 #define SPEEX_SET_VBR 12

Set VBR on (1) or off (0)

7.1.2.55 #define SPEEX_SET_VBR_MAX_BITRATE 42

Sets the max bit-rate allowed in VBR mode

7.1.2.56 #define SPEEX_SET_VBR_QUALITY 14

Set quality value for VBR encoding (0-10)

7.1.2.57 #define SPEEX_SET_WIDEBAND 105

Used internally

7.1.2.58 #define SPEEX_SUBMODE_BITS_PER_FRAME 1

Query the size of an encoded frame for a particular sub-mode

7.1.3 Typedef Documentation

7.1.3.1 typedef int(* decode_func)(void *state, SpeexBits *bits, void *out)

Main decoding function

7.1.3.2 typedef int(* decoder_ctl_func)(void *state, int request, void *ptr)

Function for controlling the decoder options

7.1.3.3 typedef void(* decoder_destroy_func)(void *st)

Decoder state destruction function

7.1.3.4 typedef void*(* decoder_init_func)(const struct SpeexMode *mode)

Decoder state initialization function

7.1.3.5 typedef int(* encode_func)(void *state, void *in, SpeexBits *bits)

Main encoding function

7.1.3.6 typedef int(* encoder_ctl_func)(void *state, int request, void *ptr)

Function for controlling the encoder options

7.1.3.7 typedef void(* encoder_destroy_func)(void *st)

Encoder state destruction function

7.1.3.8 typedef void*(* encoder_init_func)(const struct SpeexMode *mode)

Encoder state initialization function

7.1.3.9 typedef int(* mode_query_func)(const void *mode, int request, void *ptr)

Query function for a mode

7.1.4 Function Documentation

7.1.4.1 int speex_decode (void * *state*, SpeexBits * *bits*, float * *out*)

Uses an existing decoder state to decode one frame of speech from bit-stream bits. The output speech is saved written to out.

Parameters:

state Decoder state

bits Bit-stream from which to decode the frame (NULL if the packet was lost)

out Where to write the decoded frame

Returns:

return status (0 for no error, -1 for end of stream, -2 corrupt stream)

7.1.4.2 int speex_decode_int (void * *state*, SpeexBits * *bits*, spx_int16_t * *out*)

Uses an existing decoder state to decode one frame of speech from bit-stream bits. The output speech is saved written to out.

Parameters:

state Decoder state

bits Bit-stream from which to decode the frame (NULL if the packet was lost)

out Where to write the decoded frame

Returns:

return status (0 for no error, -1 for end of stream, -2 corrupt stream)

7.1.4.3 int speex_decoder_ctl (void * *state*, int *request*, void * *ptr*)

Used like the ioctl function to control the encoder parameters

Parameters:

state Decoder state

request ioctl-type request (one of the SPEEX_* macros)

ptr Data exchanged to-from function

Returns:

0 if no error, -1 if request in unknown

7.1.4.4 void speex_decoder_destroy (void * *state*)

Frees all resources associated to an existing decoder state.

Parameters:

state State to be destroyed

7.1.4.5 void* speex_decoder_init (const SpeexMode * *mode*)

Returns a handle to a newly created decoder state structure. For now, the mode argument can be &nb_mode or &wb_mode . In the future, more modes may be added. Note that for now if you have more than one channels to decode, you need one state per channel.

Parameters:

mode Speex mode (one of speex_nb_mode or speex_wb_mode)

Returns:

A newly created decoder state

7.1.4.6 int speex_encode (void * *state*, float * *in*, SpeexBits * *bits*)

Uses an existing encoder state to encode one frame of speech pointed to by "in". The encoded bit-stream is saved in "bits".

Parameters:

state Encoder state

in Frame that will be encoded with a +-2^15 range

bits Bit-stream where the data will be written

Returns:

0 if frame needs not be transmitted (DTX only), 1 otherwise

7.1.4.7 int speex_encode_int (void * *state*, spx_int16_t * *in*, SpeexBits * *bits*)

Uses an existing encoder state to encode one frame of speech pointed to by "in". The encoded bit-stream is saved in "bits".

Parameters:

state Encoder state

in Frame that will be encoded with a +-2^15 range

bits Bit-stream where the data will be written

Returns:

0 if frame needs not be transmitted (DTX only), 1 otherwise

7.1.4.8 int speex_encoder_ctl (void * *state*, int *request*, void * *ptr*)

Used like the ioctl function to control the encoder parameters

Parameters:

state Encoder state

request ioctl-type request (one of the SPEEX_* macros)

ptr Data exchanged to-from function

Returns:

0 if no error, -1 if request in unknown

7.1.4.9 void speex_encoder_destroy (void * *state*)

Frees all resources associated to an existing Speex encoder state.

Parameters:

state Encoder state to be destroyed

7.1.4.10 void* speex_encoder_init (const SpeexMode * *mode*)

Returns a handle to a newly created Speex encoder state structure. For now, the "mode" argument can be &nb_mode or &wb_mode . In the future, more modes may be added. Note that for now if you have more than one channels to encode, you need one state per channel.

Parameters:

mode The mode to use (either speex_nb_mode or speex_wb.mode)

Returns:

A newly created encoder

7.1.4.11 int speex_lib_ctl (int *request*, void * *ptr*)

Functions for controlling the behavior of libspeex

Parameters:

request ioctl-type request (one of the SPEEX_LIB_* macros)

ptr Data exchanged to-from function

7.1.4.12 const SpeexMode* speex_lib_get_mode (int *mode*)

Obtain one of the modes available

7.1.4.13 int speex_mode_query (const SpeexMode * *mode*, int *request*, void * *ptr*)

Query function for mode information

Parameters:

mode Speex mode

request ioctl-type request (one of the SPEEX_* macros)

ptr Data exchanged to-from function

7.1.5 Variable Documentation

7.1.5.1 const SpeexMode* const speex_mode_list[SPEEX_NB_MODES]

List of all modes available

7.1.5.2 const SpeexMode speex_nb_mode

Default narrowband mode

7.1.5.3 const SpeexMode speex_uwb_mode

Default "ultra-wideband" mode

7.1.5.4 const SpeexMode speex_wb_mode

Default wideband mode

7.2 speex_bits.h File Reference

Handles bit packing/unpacking.

Classes

- struct SpeexBits

Functions

- void speex_bits_init (SpeexBits *bits)
- void speex_bits_init_buffer (SpeexBits *bits, void *buff, int buf_size)
- void speex_bits_destroy (SpeexBits *bits)
- void speex_bits_reset (SpeexBits *bits)
- void speex_bits_rewind (SpeexBits *bits)
- void speex_bits_read_from (SpeexBits *bits, char *bytes, int len)
- void speex_bits_read_whole_bytes (SpeexBits *bits, char *bytes, int len)
- int speex_bits_write (SpeexBits *bits, char *bytes, int max_len)
- int speex_bits_write_whole_bytes (SpeexBits *bits, char *bytes, int max_len)
- void speex_bits_pack (SpeexBits *bits, int data, int nbBits)
- int speex_bits_unpack_signed (SpeexBits *bits, int nbBits)
- unsigned int speex_bits_unpack_unsigned (SpeexBits *bits, int nbBits)
- int speex_bits_nbytes (SpeexBits *bits)
- unsigned int speex_bits_peek_unsigned (SpeexBits *bits, int nbBits)
- int speex_bits_peek (SpeexBits *bits)
- void speex_bits_advance (SpeexBits *bits, int n)
- int speex_bits_remaining (SpeexBits *bits)
- void speex_bits_insert_terminator (SpeexBits *bits)

7.2.1 Detailed Description

Handles bit packing/unpacking.

7.2.2 Function Documentation

7.2.2.1 void speex_bits_advance (SpeexBits * *bits*, int *n*)

Advances the position of the "bit cursor" in the stream

Parameters:

bits Bit-stream to operate on

n Number of bits to advance

7.2.2.2 void speex_bits_destroy (SpeexBits * *bits*)

Frees all resources associated to a SpeexBits struct. Right now this does nothing since no resources are allocated, but this could change in the future.

7.2.2.3 void speex_bits_init (SpeexBits * *bits*)

Initializes and allocates resources for a SpeexBits struct

7.2.2.4 void speex_bits_init_buffer (SpeexBits * *bits*, void * *buff*, int *buf_size*)

Initializes SpeexBits struct using a pre-allocated buffer

7.2.2.5 void speex_bits_insert_terminator (SpeexBits * *bits*)

Insert a terminator so that the data can be sent as a packet while auto-detecting the number of frames in each packet

Parameters:
bits Bit-stream to operate on

7.2.2.6 int speex_bits_nbytes (SpeexBits * *bits*)

Returns the number of bytes in the bit-stream, including the last one even if it is not "full"

Parameters:
bits Bit-stream to operate on

Returns:
Number of bytes in the stream

7.2.2.7 void speex_bits_pack (SpeexBits * *bits*, int *data*, int *nbBits*)

Append bits to the bit-stream

Parameters:
bits Bit-stream to operate on
data Value to append as integer
nbBits number of bits to consider in "data"

7.2.2.8 int speex_bits_peek (SpeexBits * *bits*)

Get the value of the next bit in the stream, without modifying the "cursor" position

Parameters:
bits Bit-stream to operate on

7.2.2.9 unsigned int speex_bits_peek_unsigned (SpeexBits * *bits*, int *nbBits*)

Same as speex_bits_unpack_unsigned, but without modifying the cursor position

7.2.2.10 void speex_bits_read_from (SpeexBits * *bits*, char * *bytes*, int *len*)

Initializes the bit-stream from the data in an area of memory

7.2.2.11 void speex_bits_read_whole_bytes (SpeexBits * *bits*, char * *bytes*, int *len*)

Append bytes to the bit-stream

Parameters:

bits Bit-stream to operate on

bytes pointer to the bytes what will be appended

len Number of bytes of append

7.2.2.12 int speex_bits_remaining (SpeexBits * *bits*)

Returns the number of bits remaining to be read in a stream

Parameters:

bits Bit-stream to operate on

7.2.2.13 void speex_bits_reset (SpeexBits * *bits*)

Resets bits to initial value (just after initialization, erasing content)

7.2.2.14 void speex_bits_rewind (SpeexBits * *bits*)

Rewind the bit-stream to the beginning (ready for read) without erasing the content

7.2.2.15 int speex_bits_unpack_signed (SpeexBits * *bits*, int *nbBits*)

Interpret the next bits in the bit-stream as a signed integer

Parameters:

bits Bit-stream to operate on

nbBits Number of bits to interpret

Returns:

A signed integer represented by the bits read

7.2.2.16 unsigned int speex_bits_unpack_unsigned (SpeexBits * *bits*, int *nbBits*)

Interpret the next bits in the bit-stream as an unsigned integer

Parameters:

bits Bit-stream to operate on

nbBits Number of bits to interpret

Returns:

An unsigned integer represented by the bits read

7.2.2.17 int speex_bits_write (SpeexBits * *bits*, char * *bytes*, int *max_len*)

Write the content of a bit-stream to an area of memory

7.2.2.18 int speex_bits_write_whole_bytes (SpeexBits * *bits*, char * *bytes*, int *max_len*)

Like speex_bits_write, but writes only the complete bytes in the stream. Also removes the written bytes from the stream

7.3 speex_callbacks.h File Reference

Describes callback handling and in-band signalling.

`#include "speex.h"`

Classes

- struct SpeexCallback

Defines

- #define SPEEX_MAX_CALLBACKS 16
- #define SPEEX_INBAND_ENH_REQUEST 0
- #define SPEEX_INBAND_RESERVED1 1
- #define SPEEX_INBAND_MODE_REQUEST 2
- #define SPEEX_INBAND_LOW_MODE_REQUEST 3
- #define SPEEX_INBAND_HIGH_MODE_REQUEST 4
- #define SPEEX_INBAND_VBR_QUALITY_REQUEST 5
- #define SPEEX_INBAND_ACKNOWLEDGE_REQUEST 6
- #define SPEEX_INBAND_VBR_REQUEST 7
- #define SPEEX_INBAND_CHAR 8
- #define SPEEX_INBAND_STEREO 9
- #define SPEEX_INBAND_MAX_BITRATE 10
- #define SPEEX_INBAND_ACKNOWLEDGE 12

Typedefs

- typedef int(∗ speex_callback_func)(SpeexBits ∗bits, void ∗state, void ∗data)

Functions

- int speex_inband_handler (SpeexBits ∗bits, SpeexCallback ∗callback_list, void ∗state)
- int speex_std_mode_request_handler (SpeexBits ∗bits, void ∗state, void ∗data)
- int speex_std_high_mode_request_handler (SpeexBits ∗bits, void ∗state, void ∗data)
- int speex_std_char_handler (SpeexBits ∗bits, void ∗state, void ∗data)
- int speex_default_user_handler (SpeexBits ∗bits, void ∗state, void ∗data)
- int **speex_std_low_mode_request_handler** (SpeexBits ∗bits, void ∗state, void ∗data)
- int **speex_std_vbr_request_handler** (SpeexBits ∗bits, void ∗state, void ∗data)
- int **speex_std_enh_request_handler** (SpeexBits ∗bits, void ∗state, void ∗data)
- int **speex_std_vbr_quality_request_handler** (SpeexBits ∗bits, void ∗state, void ∗data)

7.3.1 Detailed Description

Describes callback handling and in-band signalling.

7.3.2 Define Documentation

7.3.2.1 #define SPEEX_INBAND_ACKNOWLEDGE 12

Acknowledge packet reception

7.3.2.2 #define SPEEX_INBAND_ACKNOWLEDGE_REQUEST 6

Request to be sent acknowledge

7.3.2.3 #define SPEEX_INBAND_CHAR 8

Send a character in-band

7.3.2.4 #define SPEEX_INBAND_ENH_REQUEST 0

Request for perceptual enhancement (1 for on, 0 for off)

7.3.2.5 #define SPEEX_INBAND_HIGH_MODE_REQUEST 4

Request for a high mode change

7.3.2.6 #define SPEEX_INBAND_LOW_MODE_REQUEST 3

Request for a low mode change

7.3.2.7 #define SPEEX_INBAND_MAX_BITRATE 10

Transmit max bit-rate allowed

7.3.2.8 #define SPEEX_INBAND_MODE_REQUEST 2

Request for a mode change

7.3.2.9 #define SPEEX_INBAND_RESERVED1 1

Reserved

7.3.2.10 #define SPEEX_INBAND_STEREO 9

Intensity stereo information

7.3.2.11 #define SPEEX_INBAND_VBR_QUALITY_REQUEST 5

Request for VBR (1 on, 0 off)

7.3.2.12 #define SPEEX_INBAND_VBR_REQUEST 7

Request for VBR (1 for on, 0 for off)

7.3.2.13 #define SPEEX_MAX_CALLBACKS 16

Total number of callbacks

7.3.3 Typedef Documentation

7.3.3.1 typedef int(* speex_callback_func)(SpeexBits *bits, void *state, void *data)

Callback function type

7.3.4 Function Documentation

7.3.4.1 int speex_default_user_handler (SpeexBits * *bits*, void * *state*, void * *data*)

Default handler for user-defined requests: in this case, just ignore

7.3.4.2 int speex_inband_handler (SpeexBits * *bits*, SpeexCallback * *callback_list*, void * *state*)

Handle in-band request

7.3.4.3 int speex_std_char_handler (SpeexBits * *bits*, void * *state*, void * *data*)

Standard handler for in-band characters (write to stderr)

7.3.4.4 int speex_std_high_mode_request_handler (SpeexBits * *bits*, void * *state*, void * *data*)

Standard handler for high mode request (change high mode, no questions asked)

7.3.4.5 int speex_std_mode_request_handler (SpeexBits * *bits*, void * *state*, void * *data*)

Standard handler for mode request (change mode, no questions asked)

7.4 speex_echo.h File Reference

Echo cancellation.

```
#include "speex/speex_types.h"
```

Defines

- #define SPEEX_ECHO_GET_FRAME_SIZE 3
- #define SPEEX_ECHO_SET_SAMPLING_RATE 24
- #define SPEEX_ECHO_GET_SAMPLING_RATE 25

Typedefs

- typedef SpeexEchoState_ **SpeexEchoState**

Functions

- SpeexEchoState * speex_echo_state_init (int frame_size, int filter_length)
- void speex_echo_state_destroy (SpeexEchoState *st)
- void speex_echo_cancel (SpeexEchoState *st, const spx_int16_t *rec, const spx_int16_t *play, spx_int16_t *out, spx_int32_t *Yout)
- void speex_echo_capture (SpeexEchoState *st, const spx_int16_t *rec, spx_int16_t *out, spx_int32_t *Yout)
- void speex_echo_playback (SpeexEchoState *st, const spx_int16_t *play)
- void speex_echo_state_reset (SpeexEchoState *st)
- int speex_echo_ctl (SpeexEchoState *st, int request, void *ptr)

7.4.1 Detailed Description

Echo cancellation.

7.4.2 Define Documentation

7.4.2.1 #define SPEEX_ECHO_GET_FRAME_SIZE 3

Obtain frame size used by the AEC

7.4.2.2 #define SPEEX_ECHO_GET_SAMPLING_RATE 25

Get sampling rate

7.4.2.3 #define SPEEX_ECHO_SET_SAMPLING_RATE 24

Set sampling rate

7.4.3 Function Documentation

7.4.3.1 void speex_echo_cancel (SpeexEchoState * *st*, const spx_int16_t * *rec*, const spx_int16_t * *play*, spx_int16_t * *out*, spx_int32_t * *Yout*)

Performs echo cancellation a frame

7.4.3.2 void speex_echo_capture (SpeexEchoState * *st*, const spx_int16_t * *rec*, spx_int16_t * *out*, spx_int32_t * *Yout*)

Perform echo cancellation using internal playback buffer

7.4.3.3 int speex_echo_ctl (SpeexEchoState * *st*, int *request*, void * *ptr*)

Used like the ioctl function to control the echo canceller parameters

Parameters:

state Encoder state

request ioctl-type request (one of the SPEEX_ECHO_* macros)

ptr Data exchanged to-from function

Returns:

0 if no error, -1 if request in unknown

7.4.3.4 void speex_echo_playback (SpeexEchoState * *st*, const spx_int16_t * *play*)

Let the echo canceller know that a frame was just played

7.4.3.5 void speex_echo_state_destroy (SpeexEchoState * *st*)

Destroys an echo canceller state

7.4.3.6 SpeexEchoState* speex_echo_state_init (int *frame_size*, int *filter_length*)

Creates a new echo canceller state

7.4.3.7 void speex_echo_state_reset (SpeexEchoState * *st*)

Reset the echo canceller state

7.5 speex_header.h File Reference

Describes the Speex header.

```
#include "speex/speex_types.h"
```

Classes

- struct SpeexHeader

Defines

- #define **SPEEX_HEADER_STRING_LENGTH** 8
- #define SPEEX_HEADER_VERSION_LENGTH 20

Functions

- void speex_init_header (SpeexHeader *header, int rate, int nb_channels, const struct SpeexMode *m)
- char * speex_header_to_packet (SpeexHeader *header, int *size)
- SpeexHeader * speex_packet_to_header (char *packet, int size)

7.5.1 Detailed Description

Describes the Speex header.

7.5.2 Define Documentation

7.5.2.1 #define SPEEX_HEADER_VERSION_LENGTH 20

Maximum number of characters for encoding the Speex version number in the header

7.5.3 Function Documentation

7.5.3.1 char* speex_header_to_packet (SpeexHeader * *header*, int * *size*)

Creates the header packet from the header itself (mostly involves endianness conversion)

7.5.3.2 void speex_init_header (SpeexHeader * *header*, int *rate*, int *nb_channels*, const struct SpeexMode * *m*)

Initializes a SpeexHeader using basic information

7.5.3.3 SpeexHeader* speex_packet_to_header (char * *packet*, int *size*)

Creates a SpeexHeader from a packet

7.6 speex_jitter.h File Reference

Adaptive jitter buffer for Speex.

```
#include "speex.h"
#include "speex_bits.h"
```

Classes

- struct **_JitterBufferPacket**
- struct SpeexJitter

Defines

- #define **JITTER_BUFFER_OK** 0
- #define **JITTER_BUFFER_MISSING** 1
- #define **JITTER_BUFFER_INCOMPLETE** 2
- #define **JITTER_BUFFER_INTERNAL_ERROR** -1
- #define **JITTER_BUFFER_BAD_ARGUMENT** -2

Typedefs

- typedef JitterBuffer_ **JitterBuffer**
- typedef _JitterBufferPacket **JitterBufferPacket**

Functions

- JitterBuffer * jitter_buffer_init (int tick)
- void jitter_buffer_reset (JitterBuffer *jitter)
- void jitter_buffer_destroy (JitterBuffer *jitter)
- void jitter_buffer_put (JitterBuffer *jitter, const JitterBufferPacket *packet)
- int jitter_buffer_get (JitterBuffer *jitter, JitterBufferPacket *packet, spx_uint32_t *current_timestamp)
- int jitter_buffer_get_pointer_timestamp (JitterBuffer *jitter)
- void jitter_buffer_tick (JitterBuffer *jitter)
- void speex_jitter_init (SpeexJitter *jitter, void *decoder, int sampling_rate)
- void speex_jitter_destroy (SpeexJitter *jitter)
- void speex_jitter_put (SpeexJitter *jitter, char *packet, int len, int timestamp)
- void speex_jitter_get (SpeexJitter *jitter, spx_int16_t *out, int *start_offset)
- int speex_jitter_get_pointer_timestamp (SpeexJitter *jitter)

7.6.1 Detailed Description

Adaptive jitter buffer for Speex.

7.6.2 Function Documentation

7.6.2.1 void jitter_buffer_destroy (JitterBuffer * *jitter*)

Destroy jitter buffer

7.6.2.2 int jitter_buffer_get (JitterBuffer * *jitter*, JitterBufferPacket * *packet*, spx_uint32_t * *current_timestamp*)

Get one packet from the jitter buffer

7.6.2.3 int jitter_buffer_get_pointer_timestamp (JitterBuffer * *jitter*)

Get pointer timestamp of jitter buffer

7.6.2.4 JitterBuffer* jitter_buffer_init (int *tick*)

Initialise jitter buffer

7.6.2.5 void jitter_buffer_put (JitterBuffer * *jitter*, const JitterBufferPacket * *packet*)

Put one packet into the jitter buffer

7.6.2.6 void jitter_buffer_reset (JitterBuffer * *jitter*)

Reset jitter buffer

7.6.2.7 void jitter_buffer_tick (JitterBuffer * *jitter*)

Advance by one tick

7.6.2.8 void speex_jitter_destroy (SpeexJitter * *jitter*)

Destroy jitter buffer

7.6.2.9 void speex_jitter_get (SpeexJitter * *jitter*, spx_int16_t * *out*, int * *start_offset*)

Get one packet from the jitter buffer

7.6.2.10 int speex_jitter_get_pointer_timestamp (SpeexJitter * *jitter*)

Get pointer timestamp of jitter buffer

7.6.2.11 void speex_jitter_init (SpeexJitter * *jitter*, void * *decoder*, int *sampling_rate*)

Initialise jitter buffer

7.6.2.12 void speex_jitter_put (SpeexJitter * *jitter*, char * *packet*, int *len*, int *timestamp*)

Put one packet into the jitter buffer

7.7 speex_noglobals.h File Reference

Dynamically allocates the different modes of the codec.

Typedefs

- typedef SpeexMode **SpeexMode**

Functions

- const SpeexMode * speex_mode_new (int modeID)
- void speex_mode_destroy (const SpeexMode *mode)

7.7.1 Detailed Description

Dynamically allocates the different modes of the codec.

7.7.2 Function Documentation

7.7.2.1 void speex_mode_destroy (const SpeexMode * *mode*)

Destroy a mode

7.7.2.2 const SpeexMode* speex_mode_new (int *modeID*)

Instantiate a mode

7.8 speex_preprocess.h File Reference

Speex preprocessor.

```
#include "speex/speex_types.h"
```

Classes

- struct SpeexPreprocessState

Defines

- #define SPEEX_PREPROCESS_SET_DENOISE 0
- #define SPEEX_PREPROCESS_GET_DENOISE 1
- #define SPEEX_PREPROCESS_SET_AGC 2
- #define SPEEX_PREPROCESS_GET_AGC 3
- #define SPEEX_PREPROCESS_SET_VAD 4
- #define SPEEX_PREPROCESS_GET_VAD 5
- #define SPEEX_PREPROCESS_SET_AGC_LEVEL 6
- #define SPEEX_PREPROCESS_GET_AGC_LEVEL 7
- #define SPEEX_PREPROCESS_SET_DEREVERB 8
- #define SPEEX_PREPROCESS_GET_DEREVERB 9
- #define SPEEX_PREPROCESS_SET_DEREVERB_LEVEL 10
- #define SPEEX_PREPROCESS_GET_DEREVERB_LEVEL 11
- #define SPEEX_PREPROCESS_SET_DEREVERB_DECAY 12
- #define SPEEX_PREPROCESS_GET_DEREVERB_DECAY 13
- #define **SPEEX_PREPROCESS_SET_PROB_START** 14
- #define **SPEEX_PREPROCESS_GET_PROB_START** 15
- #define **SPEEX_PREPROCESS_SET_PROB_CONTINUE** 16
- #define **SPEEX_PREPROCESS_GET_PROB_CONTINUE** 17

Functions

- SpeexPreprocessState * speex_preprocess_state_init (int frame_size, int sampling_rate)
- void speex_preprocess_state_destroy (SpeexPreprocessState *st)
- int speex_preprocess (SpeexPreprocessState *st, spx_int16_t *x, spx_int32_t *echo)
- void speex_preprocess_estimate_update (SpeexPreprocessState *st, spx_int16_t *x, spx_int32_t *echo)
- int speex_preprocess_ctl (SpeexPreprocessState *st, int request, void *ptr)

7.8.1 Detailed Description

Speex preprocessor.

7.8.2 Define Documentation

7.8.2.1 #define SPEEX_PREPROCESS_GET_AGC 3

Get preprocessor Automatic Gain Control state

7.8.2.2 #define SPEEX_PREPROCESS_GET_AGC_LEVEL 7

Get preprocessor Automatic Gain Control level

7.8.2.3 #define SPEEX_PREPROCESS_GET_DENOISE 1

Get preprocessor denoiser state

7.8.2.4 #define SPEEX_PREPROCESS_GET_DEREVERB 9

Get preprocessor dereverb state

7.8.2.5 #define SPEEX_PREPROCESS_GET_DEREVERB_DECAY 13

Get preprocessor dereverb decay

7.8.2.6 #define SPEEX_PREPROCESS_GET_DEREVERB_LEVEL 11

Get preprocessor dereverb level

7.8.2.7 #define SPEEX_PREPROCESS_GET_VAD 5

Get preprocessor Voice Activity Detection state

7.8.2.8 #define SPEEX_PREPROCESS_SET_AGC 2

Set preprocessor Automatic Gain Control state

7.8.2.9 #define SPEEX_PREPROCESS_SET_AGC_LEVEL 6

Set preprocessor Automatic Gain Control level

7.8.2.10 #define SPEEX_PREPROCESS_SET_DENOISE 0

Set preprocessor denoiser state

7.8.2.11 #define SPEEX_PREPROCESS_SET_DEREVERB 8

Set preprocessor dereverb state

7.8.2.12 #define SPEEX_PREPROCESS_SET_DEREVERB_DECAY 12

Set preprocessor dereverb decay

7.8.2.13 #define SPEEX_PREPROCESS_SET_DEREVERB_LEVEL 10

Set preprocessor dereverb level

7.8.2.14 #define SPEEX_PREPROCESS_SET_VAD 4

Set preprocessor Voice Activity Detection state

7.8.3 Function Documentation

7.8.3.1 int speex_preprocess (SpeexPreprocessState * *st*, spx_int16_t * *x*, spx_int32_t * *echo*)

Preprocess a frame

7.8.3.2 int speex_preprocess_ctl (SpeexPreprocessState * *st*, int *request*, void * *ptr*)

Used like the ioctl function to control the preprocessor parameters

7.8.3.3 void speex_preprocess_estimate_update (SpeexPreprocessState * *st*, spx_int16_t * *x*, spx_int32_t * *echo*)

Preprocess a frame

7.8.3.4 void speex_preprocess_state_destroy (SpeexPreprocessState * *st*)

Destroys a denoising state

7.8.3.5 SpeexPreprocessState* speex_preprocess_state_init (int *frame_size*, int *sampling_rate*)

Creates a new preprocessing state

7.9 speex_stereo.h File Reference

Describes the handling for intensity stereo.

```
#include "speex/speex_types.h"
#include "speex/speex_bits.h"
```

Classes

- struct SpeexStereoState

Defines

- #define SPEEX_STEREO_STATE_INIT {1,.5,1,1,0,0}

Functions

- void speex_encode_stereo (float *data, int frame_size, SpeexBits *bits)
- void speex_encode_stereo_int (spx_int16_t *data, int frame_size, SpeexBits *bits)
- void speex_decode_stereo (float *data, int frame_size, SpeexStereoState *stereo)
- void speex_decode_stereo_int (spx_int16_t *data, int frame_size, SpeexStereoState *stereo)
- int speex_std_stereo_request_handler (SpeexBits *bits, void *state, void *data)

7.9.1 Detailed Description

Describes the handling for intensity stereo.

7.9.2 Define Documentation

7.9.2.1 #define SPEEX_STEREO_STATE_INIT {1,.5,1,1,0,0}

Initialization value for a stereo state

7.9.3 Function Documentation

7.9.3.1 void speex_decode_stereo (float * *data*, int *frame_size*, SpeexStereoState * *stereo*)

Transforms a mono frame into a stereo frame using intensity stereo info

7.9.3.2 void speex_decode_stereo_int (spx_int16_t * *data*, int *frame_size*, SpeexStereoState * *stereo*)

Transforms a mono frame into a stereo frame using intensity stereo info

7.9.3.3 void speex_encode_stereo (float * *data*, int *frame_size*, SpeexBits * *bits*)

Transforms a stereo frame into a mono frame and stores intensity stereo info in 'bits'

7.9.3.4 void speex_encode_stereo_int (spx_int16_t * *data*, int *frame_size*, SpeexBits * *bits*)

Transforms a stereo frame into a mono frame and stores intensity stereo info in 'bits'

7.9.3.5 int speex_std_stereo_request_handler (SpeexBits * *bits*, void * *state*, void * *data*)

Callback handler for intensity stereo info

7.10 speex_types.h File Reference

Speex types.

```
#include <speex/speex_config_types.h>
```

7.10.1 Detailed Description

Speex types.

Index