Speex Reference Manual

1.2-beta2

Generated by Doxygen 1.5.1

Wed May 23 00:53:16 2007

Contents

Chapter 1

Speex Module Index

1.1 Speex Modules

Here is a list of all modules:

Chapter 2

Speex Directory Hierarchy

2.1 Speex Directories

This directory hierarchy 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 Module Documentation

5.1 Speex encoder and decoder

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_ACTIVITY 47
- #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]

5.1.1 Detailed Description

This is the Speex codec itself.

5.1.2 Define Documentation

5.1.2.1 #define SPEEX_GET_ABR 33

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

5.1.2.2 #define SPEEX_GET_ACTIVITY 47

Get "activity level" of the last decoded frame, i.e. now much damage we cause if we remove the frame

5.1.2.3 #define SPEEX_GET_BITRATE 19

Get current bit-rate used by the encoder or decoder

5.1.2.4 #define SPEEX_GET_COMPLEXITY 17

Get current complexity of the encoder (0-10)

5.1.2.5 #define SPEEX_GET_DTX 35

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

5.1.2.6 #define SPEEX_GET_ENH 1

Get enhancement state (decoder only)

5.1.2.7 #define SPEEX_GET_FRAME_SIZE 3

Obtain frame size used by encoder/decoder

5.1.2.8 #define SPEEX_GET_HIGH_MODE 11

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

5.1.2.9 #define SPEEX_GET_HIGHPASS 45

Get status of input/output high-pass filtering

5.1.2.10 #define SPEEX_GET_LOOKAHEAD 39

Returns the lookahead used by Speex

5.1.2.11 #define SPEEX_GET_LOW_MODE 9

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

5.1.2.12 #define SPEEX_GET_MODE 7

Get current sub-mode in use

5.1.2.13 #define SPEEX_GET_PF 1

Equivalent to SPEEX_GET_ENH

5.1.2.14 #define SPEEX_GET_PLC_TUNING 41

Gets tuning for PLC

5.1.2.15 #define SPEEX_GET_RELATIVE_QUALITY 29

Get VBR info (mostly used internally)

5.1.2.16 #define SPEEX_GET_SAMPLING_RATE 25

Get sampling rate used in bit-rate computation

5.1.2.17 #define SPEEX_GET_SUBMODE_ENCODING 37

Get submode encoding in each frame

5.1.2.18 #define SPEEX_GET_VAD 31

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

5.1.2.19 #define SPEEX_GET_VBR 13

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

5.1.2.20 #define SPEEX_GET_VBR_MAX_BITRATE 43

Gets the max bit-rate allowed in VBR mode

5.1.2.21 #define SPEEX_GET_VBR_QUALITY 15

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

5.1.2.22 #define SPEEX_LIB_GET_EXTRA_VERSION 7

Get extra Speex version

5.1.2.23 #define SPEEX_LIB_GET_MAJOR_VERSION 1

Get major Speex version

5.1.2.24 #define SPEEX_LIB_GET_MICRO_VERSION 5

Get micro Speex version

5.1.2.25 #define SPEEX_LIB_GET_MINOR_VERSION 3

Get minor Speex version

5.1.2.26 #define SPEEX_LIB_GET_VERSION_STRING 9

Get Speex version string

5.1.2.27 #define SPEEX_MODE_FRAME_SIZE 0

Query the frame size of a mode

5.1.2.28 #define SPEEX_MODEID_NB 0

modeID for the defined narrowband mode

5.1.2.29 #define SPEEX_MODEID_UWB 2

modeID for the defined ultra-wideband mode

5.1.2.30 #define SPEEX_MODEID_WB 1

modeID for the defined wideband mode

5.1.2.31 #define SPEEX_NB_MODES 3

Number of defined modes in Speex

5.1.2.32 #define SPEEX_RESET_STATE 26

Reset the encoder/decoder memories to zero

5.1.2.33 #define SPEEX_SET_ABR 32

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

5.1.2.34 #define SPEEX_SET_BITRATE 18

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

5.1.2.35 #define SPEEX_SET_COMPLEXITY 16

Set complexity of the encoder (0-10)

5.1.2.36 #define SPEEX_SET_DTX 34

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

5.1.2.37 #define SPEEX_SET_ENH 0

Set enhancement on/off (decoder only)

5.1.2.38 #define SPEEX_SET_HANDLER 20

Define a handler function for in-band Speex request

5.1.2.39 #define SPEEX_SET_HIGH_MODE 10

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

5.1.2.40 #define SPEEX_SET_HIGHPASS 44

Turn on/off input/output high-pass filtering

5.1.2.41 #define SPEEX_SET_LOW_MODE 8

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

5.1.2.42 #define SPEEX_SET_MODE 6

Set sub-mode to use

5.1.2.43 #define SPEEX_SET_PF 0

Equivalent to SPEEX_SET_ENH

5.1.2.44 #define SPEEX_SET_PLC_TUNING 40

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

5.1.2.45 #define SPEEX_SET_QUALITY 4

Set quality value

5.1.2.46 #define SPEEX_SET_SAMPLING_RATE 24

Set sampling rate used in bit-rate computation

5.1.2.47 #define SPEEX_SET_SUBMODE_ENCODING 36

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

5.1.2.48 #define SPEEX_SET_USER_HANDLER 22

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

5.1.2.49 #define SPEEX_SET_VAD 30

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

5.1.2.50 #define SPEEX_SET_VBR 12

Set VBR on (1) or off (0)

5.1.2.51 #define SPEEX_SET_VBR_MAX_BITRATE 42

Sets the max bit-rate allowed in VBR mode

5.1.2.52 #define SPEEX_SET_VBR_QUALITY 14

Set quality value for VBR encoding (0-10)

5.1.2.53 #define SPEEX_SUBMODE_BITS_PER_FRAME 1

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

5.1.3 Typedef Documentation

5.1.3.1 typedef int(∗) decode_func(void ∗state, SpeexBits ∗bits, void ∗out)

Main decoding function

5.1.3.2 typedef int(∗) decoder_ctl_func(void ∗state, int request, void ∗ptr)

Function for controlling the decoder options

5.1.3.3 typedef void(∗) decoder_destroy_func(void ∗st)

Decoder state destruction function

5.1.3.4 typedef void∗(∗) decoder_init_func(const struct SpeexMode ∗mode)

Decoder state initialization function

5.1.3.5 typedef int(∗) encode_func(void ∗state, void ∗in, SpeexBits ∗bits)

Main encoding function

5.1.3.6 typedef int(∗) encoder_ctl_func(void ∗state, int request, void ∗ptr)

Function for controlling the encoder options

5.1.3.7 typedef void(∗) encoder_destroy_func(void ∗st)

Encoder state destruction function

5.1.3.8 typedef void∗(∗) encoder_init_func(const struct SpeexMode ∗mode)

Encoder state initialization function

5.1.3.9 typedef int(∗) mode_query_func(const void ∗mode, int request, void ∗ptr)

Query function for a mode

5.1.4 Function Documentation

5.1.4.1 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)

5.1.4.2 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)

5.1.4.3 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, -2 for invalid parameter

5.1.4.4 void speex_decoder_destroy (void ∗ *state*)

Frees all resources associated to an existing decoder state.

Parameters:

state State to be destroyed

5.1.4.5 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 or NULL if state allocation fails

5.1.4.6 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. This data MAY be overwritten by the encoder and should be considered uninitialised after the call.

bits Bit-stream where the data will be written

Returns:

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

5.1.4.7 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

5.1.4.8 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, -2 for invalid parameter

5.1.4.9 void speex_encoder_destroy (void ∗ *state*)

Frees all resources associated to an existing Speex encoder state.

Parameters:

state Encoder state to be destroyed

5.1.4.10 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 state or NULL if state allocation fails

5.1.4.11 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

Returns:

0 if no error, -1 if request in unknown, -2 for invalid parameter

5.1.4.12 const SpeexMode∗ speex_lib_get_mode (int *mode*)

Obtain one of the modes available

5.1.4.13 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

Returns:

0 if no error, -1 if request in unknown, -2 for invalid parameter

5.1.5 Variable Documentation

5.1.5.1 const SpeexMode* const speex_mode_list[SPEEX_NB_MODES]

List of all modes available

5.1.5.2 const SpeexMode speex_nb_mode

Default narrowband mode

5.1.5.3 const SpeexMode speex_uwb_mode

Default "ultra-wideband" mode

5.1.5.4 const SpeexMode speex_wb_mode

Default wideband mode

5.2 SpeexBits: Bit-stream manipulations

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_set_bit_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)

5.2.1 Detailed Description

This is the structure that holds the bit-stream when encoding or decoding with Speex. It allows some manipulations as well.

5.2.2 Function Documentation

5.2.2.1 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

5.2.2.2 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.

5.2.2.3 void speex_bits_init (SpeexBits * *bits*)

Initializes and allocates resources for a SpeexBits struct

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

Initializes SpeexBits struct using a pre-allocated buffer

5.2.2.5 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

5.2.2.6 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

5.2.2.7 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"

5.2.2.8 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

Returns:

Value of the bit peeked (one bit only)

5.2.2.9 unsigned int speex_bits_peek_unsigned (SpeexBits ∗ *bits*, int *nbBits*)

Same as speex_bits_unpack_unsigned, but without modifying the cursor position

Parameters:

bits Bit-stream to operate on

nbBits Number of bits to look for

Returns:

Value of the bits peeked, interpreted as unsigned

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

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

5.2.2.11 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

5.2.2.12 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

Returns:

Number of bits that can still be read from the stream

5.2.2.13 void speex_bits_reset (SpeexBits * *bits*)

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

5.2.2.14 void speex_bits_rewind (SpeexBits * *bits*)

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

5.2.2.15 void speex_bits_set_bit_buffer (SpeexBits * *bits*, void * *buff*, int *buf_size*)

Sets the bits in a SpeexBits struct to use data from an existing buffer (for decoding without copying data)

5.2.2.16 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

5.2.2.17 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

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

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

Parameters:

bits Bit-stream to operate on

bytes Memory location where to write the bits

max_len Maximum number of bytes to write (i.e. size of the "bytes" buffer)

Returns:

Number of bytes written to the "bytes" buffer

5.2.2.19 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

5.3 Various definitions for Speex callbacks supported by the decoder.

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)

5.3.1 Define Documentation

5.3.1.1 #define SPEEX_INBAND_ACKNOWLEDGE 12

Acknowledge packet reception

5.3.1.2 #define SPEEX_INBAND_ACKNOWLEDGE_REQUEST 6

Request to be sent acknowledge

5.3.1.3 #define SPEEX_INBAND_CHAR 8

Send a character in-band

5.3.1.4 #define SPEEX_INBAND_ENH_REQUEST 0

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

5.3.1.5 #define SPEEX_INBAND_HIGH_MODE_REQUEST 4

Request for a high mode change

5.3.1.6 #define SPEEX_INBAND_LOW_MODE_REQUEST 3

Request for a low mode change

5.3.1.7 #define SPEEX_INBAND_MAX_BITRATE 10

Transmit max bit-rate allowed

5.3.1.8 #define SPEEX_INBAND_MODE_REQUEST 2

Request for a mode change

5.3.1.9 #define SPEEX_INBAND_RESERVED1 1

Reserved

5.3.1.10 #define SPEEX_INBAND_STEREO 9

Intensity stereo information

5.3.1.11 #define SPEEX_INBAND_VBR_QUALITY_REQUEST 5

Request for VBR (1 on, 0 off)

5.3.1.12 #define SPEEX_INBAND_VBR_REQUEST 7

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

5.3.1.13 #define SPEEX_MAX_CALLBACKS 16

Total number of callbacks

5.3.2 Typedef Documentation

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

Callback function type

5.3.3 Function Documentation

5.3.3.1 int speex_default_user_handler (SpeexBits ∗ *bits*, void ∗ *state*, void ∗ *data*)

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

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

Handle in-band request

5.3.3.3 int speex_std_char_handler (SpeexBits ∗ *bits*, void ∗ *state*, void ∗ *data*)

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

5.3.3.4 int speex_std_enh_request_handler (SpeexBits ∗ *bits*, void ∗ *state*, void ∗ *data*)

Standard handler for enhancer request (Turn ehnancer on/off, no questions asked)

5.3.3.5 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)

5.3.3.6 int speex_std_low_mode_request_handler (SpeexBits ∗ *bits*, void ∗ *state*, void ∗ *data*)

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

5.3.3.7 int speex_std_mode_request_handler (SpeexBits ∗ *bits*, void ∗ *state*, void ∗ *data*)

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

5.3.3.8 int speex_std_vbr_quality_request_handler (SpeexBits * *bits*, void * *state*, void * *data*)

Standard handler for VBR quality request (Set VBR quality, no questions asked)

5.3.3.9 int speex_std_vbr_request_handler (SpeexBits * *bits*, void * *state*, void * *data*)

Standard handler for VBR request (Set VBR, no questions asked)

5.4 SpeexEchoState: Acoustic echo canceller

Classes

- class SpeexEchoState

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_cancellation (SpeexEchoState *st, const spx_int16_t *rec, const spx_int16_t *play, spx_int16_t *out)
- 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)
- 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)

5.4.1 Detailed Description

This is the acoustic echo canceller module.

5.4.2 Define Documentation

5.4.2.1 #define SPEEX_ECHO_GET_FRAME_SIZE 3

Obtain frame size used by the AEC

5.4.2.2 #define SPEEX_ECHO_GET_SAMPLING_RATE 25

Get sampling rate

5.4.2.3 #define SPEEX_ECHO_SET_SAMPLING_RATE 24

Set sampling rate

5.4.3 Typedef Documentation

5.4.3.1 typedef struct SpeexEchoState_ SpeexEchoState

Internal echo canceller state. Should never be accessed directly.

5.4.4 Function Documentation

5.4.4.1 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 (deprecated)

5.4.4.2 void speex_echo_cancellation (SpeexEchoState ∗ *st*, const spx_int16_t ∗ *rec*, const spx_int16_t ∗ *play*, spx_int16_t ∗ *out*)

Performs echo cancellation a frame, based on the audio sent to the speaker (no delay is added to playback ni this form)

Parameters:

st Echo canceller state

rec signal from the microphone (near end + far end echo)

play Signal played to the speaker (received from far end)

out Returns near-end signal with echo removed

5.4.4.3 void speex_echo_capture (SpeexEchoState ∗ *st*, const spx_int16_t ∗ *rec*, spx_int16_t ∗ *out*)

Perform echo cancellation using internal playback buffer, which is delayed by two frames to account for the delay introduced by most soundcards (but it could be off!)

Parameters:

st Echo canceller state

rec signal from the microphone (near end + far end echo)

out Returns near-end signal with echo removed

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

Used like the ioctl function to control the echo canceller parameters

Parameters:

st Echo canceller 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

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

Let the echo canceller know that a frame was just queued to the soundcard

Parameters:

st Echo canceller state

play Signal played to the speaker (received from far end)

5.4.4.6 void speex_echo_state_destroy (SpeexEchoState * *st*)

Destroys an echo canceller state

Parameters:

st Echo canceller state

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

Creates a new echo canceller state

Parameters:

frame_size Number of samples to process at one time (should correspond to 10-20 ms)

filter_length Number of samples of echo to cancel (should generally correspond to 100-500 ms)

Returns:

Newly-created echo canceller state

5.4.4.8 void speex_echo_state_reset (SpeexEchoState * *st*)

Reset the echo canceller to its original state

Parameters:

st Echo canceller state

5.5 SpeexHeader: Makes it easy to write/parse an Ogg/Speex header

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)

5.5.1 Detailed Description

This is the Speex header for the Ogg encapsulation. You don't need that if you just use RTP.

5.5.2 Define Documentation

5.5.2.1 #define SPEEX_HEADER_STRING_LENGTH 8

Length of the Speex header identifier

5.5.2.2 #define SPEEX_HEADER_VERSION_LENGTH 20

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

5.5.3 Function Documentation

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

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

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

Initializes a SpeexHeader using basic information

5.5.3.3 SpeexHeader∗ speex_packet_to_header (char ∗ *packet*, int *size*)

Creates a SpeexHeader from a packet

5.6 JitterBuffer: Adaptive jitter buffer

Classes

- struct _JitterBufferPacket

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
- #define JITTER_BUFFER_SET_MARGIN 0
- #define JITTER_BUFFER_GET_MARGIN 1
- #define JITTER_BUFFER_GET_AVALIABLE_COUNT 3
- #define **JITTER_BUFFER_ADJUST_INTERPOLATE** -1
- #define **JITTER_BUFFER_ADJUST_OK** 0
- #define **JITTER_BUFFER_ADJUST_DROP** 1

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_int32_t ∗start_offset)
- int jitter_buffer_get_pointer_timestamp (JitterBuffer ∗jitter)
- void jitter_buffer_tick (JitterBuffer ∗jitter)
- int jitter_buffer_ctl (JitterBuffer ∗jitter, int request, void ∗ptr)
- int **jitter_buffer_update_delay** (JitterBuffer ∗jitter, JitterBufferPacket ∗packet, spx_int32_-t ∗start_offset)

5.6.1 Detailed Description

This is the jitter buffer that reorders UDP/RTP packets and adjusts the buffer size to maintain good quality and low latency.

5.6.2 Define Documentation

5.6.2.1 #define JITTER_BUFFER_BAD_ARGUMENT -2

Invalid argument

5.6.2.2 #define JITTER_BUFFER_GET_AVALIABLE_COUNT 3

Get the amount of avaliable packets currently buffered

5.6.2.3 #define JITTER_BUFFER_GET_MARGIN 1

Get minimum amount of extra buffering required (margin)

5.6.2.4 #define JITTER_BUFFER_INCOMPLETE 2

Packet is incomplete (does not cover the entive tick

5.6.2.5 #define JITTER_BUFFER_INTERNAL_ERROR -1

There was an error in the jitter buffer

5.6.2.6 #define JITTER_BUFFER_MISSING 1

Packet is missing

5.6.2.7 #define JITTER_BUFFER_OK 0

Packet has been retrieved

5.6.2.8 #define JITTER_BUFFER_SET_MARGIN 0

Set minimum amount of extra buffering required (margin)

5.6.3 Typedef Documentation

5.6.3.1 typedef struct JitterBuffer_ JitterBuffer

Generic adaptive jitter buffer state

5.6.3.2 typedef struct _JitterBufferPacket JitterBufferPacket

Definition of an incoming packet

5.6.4 Function Documentation

5.6.4.1 int jitter_buffer_ctl (JitterBuffer * *jitter*, int *request*, void * *ptr*)

Used like the ioctl function to control the jitter buffer parameters

Parameters:

jitter Jitter buffer state

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

ptr Data exchanged to-from function

Returns:

0 if no error, -1 if request in unknown

5.6.4.2 void jitter_buffer_destroy (JitterBuffer * *jitter*)

Destroys jitter buffer

Parameters:

jitter Jitter buffer state

5.6.4.3 int jitter_buffer_get (JitterBuffer * *jitter*, JitterBufferPacket * *packet*, spx_int32_t * *start_offset*)

Get one packet from the jitter buffer

Parameters:

jitter Jitter buffer state

packet Returned packet

current_timestamp Timestamp for the returned packet

5.6.4.4 int jitter_buffer_get_pointer_timestamp (JitterBuffer * *jitter*)

Get pointer timestamp of jitter buffer

Parameters:

jitter Jitter buffer state

5.6.4.5 JitterBuffer* jitter_buffer_init (int *tick*)

Initialises jitter buffer

Parameters:

tick Number of samples per "tick", i.e. the time period of the elements that will be retrieved

Returns:

Newly created jitter buffer state

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

Put one packet into the jitter buffer

Parameters:

jitter Jitter buffer state

packet Incoming packet

5.6.4.7 void jitter_buffer_reset (JitterBuffer * *jitter*)

Restores jitter buffer to its original state

Parameters:

jitter Jitter buffer state

5.6.4.8 void jitter_buffer_tick (JitterBuffer * *jitter*)

Advance by one tick

Parameters:

jitter Jitter buffer state

5.7 SpeexJitter: Adaptive jitter buffer specifically for Speex

Classes

- struct SpeexJitter

Functions

- 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)

5.7.1 Detailed Description

This is the jitter buffer that reorders UDP/RTP packets and adjusts the buffer size to maintain good quality and low latency. This is a simplified version that works only with Speex, but is much easier to use.

5.7.2 Function Documentation

5.7.2.1 void speex_jitter_destroy (SpeexJitter ∗ *jitter*)

Destroy jitter buffer

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

Get one packet from the jitter buffer

5.7.2.3 int speex_jitter_get_pointer_timestamp (SpeexJitter ∗ *jitter*)

Get pointer timestamp of jitter buffer

5.7.2.4 void speex_jitter_init (SpeexJitter ∗ *jitter*, void ∗ *decoder*, int *sampling_rate*)

Initialise jitter buffer

Parameters:

jitter State of the Speex jitter buffer
decoder Speex decoder to call
sampling_rate Sampling rate used by the decoder

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

Put one packet into the jitter buffer

5.8 SpeexPreprocessState: The Speex preprocessor

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
- #define SPEEX_PREPROCESS_SET_NOISE_SUPPRESS 18
- #define SPEEX_PREPROCESS_GET_NOISE_SUPPRESS 19
- #define SPEEX_PREPROCESS_SET_ECHO_SUPPRESS 20
- #define SPEEX_PREPROCESS_GET_ECHO_SUPPRESS 21
- #define SPEEX_PREPROCESS_SET_ECHO_SUPPRESS_ACTIVE 22
- #define SPEEX_PREPROCESS_GET_ECHO_SUPPRESS_ACTIVE 23
- #define SPEEX_PREPROCESS_SET_ECHO_STATE 24
- #define SPEEX_PREPROCESS_GET_ECHO_STATE 25
- #define SPEEX_PREPROCESS_SET_AGC_INCREMENT 26
- #define SPEEX_PREPROCESS_GET_AGC_INCREMENT 27
- #define SPEEX_PREPROCESS_SET_AGC_DECREMENT 28
- #define SPEEX_PREPROCESS_GET_AGC_DECREMENT 29
- #define SPEEX_PREPROCESS_SET_AGC_MAX_GAIN 30
- #define SPEEX_PREPROCESS_GET_AGC_MAX_GAIN 31

Typedefs

- typedef SpeexPreprocessState_ SpeexPreprocessState

Functions

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

5.8.1 Detailed Description

This is the Speex preprocessor. The preprocess can do noise suppression, residual echo suppression (after using the echo canceller), automatic gain control (AGC) and voice activity detection (VAD).

5.8.2 Define Documentation

5.8.2.1 #define SPEEX_PREPROCESS_GET_AGC 3

Get preprocessor Automatic Gain Control state

5.8.2.2 #define SPEEX_PREPROCESS_GET_AGC_DECREMENT 29

Get maximal gain decrease in dB/second (int32)

5.8.2.3 #define SPEEX_PREPROCESS_GET_AGC_INCREMENT 27

Get maximal gain increase in dB/second (int32)

5.8.2.4 #define SPEEX_PREPROCESS_GET_AGC_LEVEL 7

Get preprocessor Automatic Gain Control level

5.8.2.5 #define SPEEX_PREPROCESS_GET_AGC_MAX_GAIN 31

Get maximal gain in dB (int32)

5.8.2.6 #define SPEEX_PREPROCESS_GET_DENOISE 1

Get preprocessor denoiser state

5.8.2.7 #define SPEEX_PREPROCESS_GET_DEREVERB 9

Get preprocessor dereverb state

5.8.2.8 #define SPEEX_PREPROCESS_GET_DEREVERB_DECAY 13

Get preprocessor dereverb decay

5.8.2.9 #define SPEEX_PREPROCESS_GET_DEREVERB_LEVEL 11

Get preprocessor dereverb level

5.8.2.10 #define SPEEX_PREPROCESS_GET_ECHO_STATE 25

Get the corresponding echo canceller state

5.8.2.11 #define SPEEX_PREPROCESS_GET_ECHO_SUPPRESS 21

Get maximum attenuation of the residual echo in dB (negative number)

5.8.2.12 #define SPEEX_PREPROCESS_GET_ECHO_SUPPRESS_ACTIVE 23

Get maximum attenuation of the residual echo in dB when near end is active (negative number)

5.8.2.13 #define SPEEX_PREPROCESS_GET_NOISE_SUPPRESS 19

Get maximum attenuation of the noise in dB (negative number)

5.8.2.14 #define SPEEX_PREPROCESS_GET_PROB_CONTINUE 17

Get probability required for the VAD to stay in the voice state (integer percent)

5.8.2.15 #define SPEEX_PREPROCESS_GET_PROB_START 15

Get probability required for the VAD to go from silence to voice

5.8.2.16 #define SPEEX_PREPROCESS_GET_VAD 5

Get preprocessor Voice Activity Detection state

5.8.2.17 #define SPEEX_PREPROCESS_SET_AGC 2

Set preprocessor Automatic Gain Control state

5.8.2.18 #define SPEEX_PREPROCESS_SET_AGC_DECREMENT 28

Set maximal gain decrease in dB/second (int32)

5.8.2.19 #define SPEEX_PREPROCESS_SET_AGC_INCREMENT 26

Set maximal gain increase in dB/second (int32)

5.8.2.20 #define SPEEX_PREPROCESS_SET_AGC_LEVEL 6

Set preprocessor Automatic Gain Control level

5.8.2.21 #define SPEEX_PREPROCESS_SET_AGC_MAX_GAIN 30

Set maximal gain in dB (int32)

5.8.2.22 #define SPEEX_PREPROCESS_SET_DENOISE 0

Set preprocessor denoiser state

5.8.2.23 #define SPEEX_PREPROCESS_SET_DEREVERB 8

Set preprocessor dereverb state

5.8.2.24 #define SPEEX_PREPROCESS_SET_DEREVERB_DECAY 12

Set preprocessor dereverb decay

5.8.2.25 #define SPEEX_PREPROCESS_SET_DEREVERB_LEVEL 10

Set preprocessor dereverb level

5.8.2.26 #define SPEEX_PREPROCESS_SET_ECHO_STATE 24

Set the corresponding echo canceller state so that residual echo suppression can be performed (NULL for no residual echo suppression)

5.8.2.27 #define SPEEX_PREPROCESS_SET_ECHO_SUPPRESS 20

Set maximum attenuation of the residual echo in dB (negative number)

5.8.2.28 #define SPEEX_PREPROCESS_SET_ECHO_SUPPRESS_ACTIVE 22

Set maximum attenuation of the residual echo in dB when near end is active (negative number)

5.8.2.29 #define SPEEX_PREPROCESS_SET_NOISE_SUPPRESS 18

Set maximum attenuation of the noise in dB (negative number)

5.8.2.30 #define SPEEX_PREPROCESS_SET_PROB_CONTINUE 16

Set probability required for the VAD to stay in the voice state (integer percent)

5.8.2.31 #define SPEEX_PREPROCESS_SET_PROB_START 14

Set probability required for the VAD to go from silence to voice

5.8.2.32 #define SPEEX_PREPROCESS_SET_VAD 4

Set preprocessor Voice Activity Detection state

5.8.3 Typedef Documentation

5.8.3.1 typedef struct SpeexPreprocessState_ SpeexPreprocessState

State of the preprocessor (one per channel). Should never be accessed directly.

5.8.4 Function Documentation

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

Preprocess a frame (deprecated, use speex_preprocess_run() instead)

5.8.4.2 int speex_preprocess_ctl (SpeexPreprocessState ∗ *st*, int *request*, void ∗ *ptr*)

Used like the ioctl function to control the preprocessor parameters

Parameters:

st Preprocessor state

request ioctl-type request (one of the SPEEX_PREPROCESS_∗ macros)

ptr Data exchanged to-from function

Returns:

0 if no error, -1 if request in unknown

5.8.4.3 void speex_preprocess_estimate_update (SpeexPreprocessState ∗ *st*, spx_int16_t ∗ *x*)

Update preprocessor state, but do not compute the output

Parameters:

st Preprocessor state

x Audio sample vector (in only). Must be same size as specified in speex_preprocess_state_init().

5.8.4.4 int speex_preprocess_run (SpeexPreprocessState ∗ *st*, spx_int16_t ∗ *x*)

Preprocess a frame

Parameters:

st Preprocessor state

x Audio sample vector (in and out). Must be same size as specified in speex_preprocess_state_init().

Returns:

Bool value for voice activity (1 for speech, 0 for noise/silence), ONLY if VAD turned on.

5.8.4.5 void speex_preprocess_state_destroy (SpeexPreprocessState ∗ *st*)

Destroys a preprocessor state

Parameters:

st Preprocessor state to destroy

5.8.4.6 SpeexPreprocessState∗ speex_preprocess_state_init (int *frame_size*, int *sampling_rate*)

Creates a new preprocessing state. You MUST create one state per channel processed.

Parameters:

frame_size Number of samples to process at one time (should correspond to 10-20 ms). Must be the same value as that used for the echo canceller for residual echo cancellation to work.

sampling_rate Sampling rate used for the input.

Returns:

Newly created preprocessor state

5.9 SpeexStereoState: Handling Speex stereo files

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)

5.9.1 Detailed Description

This describes the Speex intensity stereo encoding/decoding

5.9.2 Define Documentation

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

Initialization value for a stereo state

5.9.3 Function Documentation

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

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

5.9.3.2 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

5.9.3.3 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'

5.9.3.4 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'

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

Callback handler for intensity stereo info

Chapter 6

Speex Directory Documentation

6.1 include/ Directory Reference

Directories

- directory speex

6.2 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_preprocess.h

  Speex preprocessor. The preprocess can do noise suppression, residual echo suppression (after using the echo canceller), automatic gain control (AGC) and voice activity detection (VAD).

- file **speex_resampler.h**
- file speex_stereo.h

  Describes the handling for intensity stereo.

- file speex_types.h

  Speex types.

Chapter 7

Speex Class Documentation

7.1 _JitterBufferPacket Struct Reference

`#include <speex_jitter.h>`

Public Attributes

- char ∗ data
- spx_uint32_t len
- spx_uint32_t timestamp
- spx_uint32_t span

7.1.1 Detailed Description

Definition of an incoming packet

7.1.2 Member Data Documentation

7.1.2.1 char∗ _JitterBufferPacket::data

Data bytes contained in the packet

7.1.2.2 spx_uint32_t _JitterBufferPacket::len

Length of the packet in bytes

7.1.2.3 spx_uint32_t _JitterBufferPacket::timestamp

Timestamp for the packet

7.1.2.4 spx_uint32_t _JitterBufferPacket::span

Time covered by the packet (same units as timestamp)

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

- speex_jitter.h

7.2 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

7.2.1 Detailed Description

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

7.2.2 Member Data Documentation

7.2.2.1 char* SpeexBits::chars

"raw" data

7.2.2.2 int SpeexBits::nbBits

Total number of bits stored in the stream

7.2.2.3 int SpeexBits::charPtr

Position of the byte "cursor"

7.2.2.4 int SpeexBits::bitPtr

Position of the bit "cursor" within the current char

7.2.2.5 int SpeexBits::owner

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

7.2.2.6 int SpeexBits::overflow

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

7.2.2.7 int SpeexBits::buf_size

Allocated size for buffer

7.2.2.8 int SpeexBits::reserved1

Reserved for future use

7.2.2.9 void* SpeexBits::reserved2

Reserved for future use

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

- speex_bits.h

7.3 SpeexCallback Struct Reference

```
#include <speex_callbacks.h>
```

Public Attributes

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

7.3.1 Detailed Description

Callback information

7.3.2 Member Data Documentation

7.3.2.1 int SpeexCallback::callback_id

ID associated to the callback

7.3.2.2 speex_callback_func SpeexCallback::func

Callback handler function

7.3.2.3 void* SpeexCallback::data

Data that will be sent to the handler

7.3.2.4 void* SpeexCallback::reserved1

Reserved for future use

7.3.2.5 int SpeexCallback::reserved2

Reserved for future use

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

- speex_callbacks.h

7.4 SpeexEchoState Class Reference

`#include <speex_echo.h>`

7.4.1 Detailed Description

This holds the state of the echo canceller. You need one per channel.

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

- speex_echo.h

7.5 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

7.5.1 Detailed Description

Speex header info for file-based formats

7.5.2 Member Data Documentation

7.5.2.1 char SpeexHeader::speex_string[SPEEX_HEADER_STRING_LENGTH]

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

7.5.2.2 char SpeexHeader::speex_version[SPEEX_HEADER_VERSION_LENGTH]

Speex version

7.5.2.3 spx_int32_t SpeexHeader::speex_version_id

Version for Speex (for checking compatibility)

7.5.2.4 spx_int32_t SpeexHeader::header_size

Total size of the header (sizeof(SpeexHeader))

7.5.2.5 spx_int32_t SpeexHeader::rate

Sampling rate used

7.5.2.6 spx_int32_t SpeexHeader::mode

Mode used (0 for narrowband, 1 for wideband)

7.5.2.7 spx_int32_t SpeexHeader::mode_bitstream_version

Version ID of the bit-stream

7.5.2.8 spx_int32_t SpeexHeader::nb_channels

Number of channels encoded

7.5.2.9 spx_int32_t SpeexHeader::bitrate

Bit-rate used

7.5.2.10 spx_int32_t SpeexHeader::frame_size

Size of frames

7.5.2.11 spx_int32_t SpeexHeader::vbr

1 for a VBR encoding, 0 otherwise

7.5.2.12 spx_int32_t SpeexHeader::frames_per_packet

Number of frames stored per Ogg packet

7.5.2.13 spx_int32_t SpeexHeader::extra_headers

Number of additional headers after the comments

7.5.2.14 spx_int32_t SpeexHeader::reserved1

Reserved for future use, must be zero

7.5.2.15 spx_int32_t SpeexHeader::reserved2

Reserved for future use, must be zero

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

- speex_header.h

7.6 SpeexJitter Struct Reference

`#include <speex_jitter.h>`

Public Attributes

- SpeexBits current_packet
- int valid_bits
- JitterBuffer * packets
- void * dec
- spx_int32_t frame_size

7.6.1 Detailed Description

Speex jitter-buffer state. Never use it directly!

7.6.2 Member Data Documentation

7.6.2.1 SpeexBits SpeexJitter::current_packet

Current Speex packet

7.6.2.2 int SpeexJitter::valid_bits

True if Speex bits are valid

7.6.2.3 JitterBuffer* SpeexJitter::packets

Generic jitter buffer state

7.6.2.4 void* SpeexJitter::dec

Pointer to Speex decoder

7.6.2.5 spx_int32_t SpeexJitter::frame_size

Frame size of Speex decoder

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

- speex_jitter.h

7.7 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

7.7.1 Detailed Description

Struct defining a Speex mode

7.7.2 Member Data Documentation

7.7.2.1 const void* SpeexMode::mode

Pointer to the low-level mode data

7.7.2.2 mode_query_func SpeexMode::query

Pointer to the mode query function

7.7.2.3 const char* SpeexMode::modeName

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

7.7.2.4 int SpeexMode::modeID

ID of the mode

7.7.2.5 int SpeexMode::bitstream_version

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

7.7.2.6 encoder_init_func SpeexMode::enc_init

Pointer to encoder initialization function

7.7.2.7 encoder_destroy_func SpeexMode::enc_destroy

Pointer to encoder destruction function

7.7.2.8 encode_func SpeexMode::enc

Pointer to frame encoding function

7.7.2.9 decoder_init_func SpeexMode::dec_init

Pointer to decoder initialization function

7.7.2.10 decoder_destroy_func SpeexMode::dec_destroy

Pointer to decoder destruction function

7.7.2.11 decode_func SpeexMode::dec

Pointer to frame decoding function

7.7.2.12 encoder_ctl_func SpeexMode::enc_ctl

ioctl-like requests for encoder

7.7.2.13 decoder_ctl_func SpeexMode::dec_ctl

ioctl-like requests for decoder

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

- speex.h

7.8 SpeexStereoState Struct Reference

```
#include <speex_stereo.h>
```

Public Attributes

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

7.8.1 Detailed Description

State used for decoding (intensity) stereo information

7.8.2 Member Data Documentation

7.8.2.1 float SpeexStereoState::balance

Left/right balance info

7.8.2.2 float SpeexStereoState::e_ratio

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

7.8.2.3 float SpeexStereoState::smooth_left

Smoothed left channel gain

7.8.2.4 float SpeexStereoState::smooth_right

Smoothed right channel gain

7.8.2.5 float SpeexStereoState::reserved1

Reserved for future use

7.8.2.6 float SpeexStereoState::reserved2

Reserved for future use

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

- speex_stereo.h

Chapter 8

Speex File Documentation

8.1 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_ACTIVITY 47
- #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]

8.1.1 Detailed Description

Describes the different modes of the codec.

8.2 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_set_bit_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)

8.2.1 Detailed Description

Handles bit packing/unpacking.

8.3 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)

8.3.1 Detailed Description

Describes callback handling and in-band signalling.

8.4 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_cancellation (SpeexEchoState *st, const spx_int16_t *rec, const spx_int16_t *play, spx_int16_t *out)
- 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)
- 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)

8.4.1 Detailed Description

Echo cancellation.

8.5 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)

8.5.1 Detailed Description

Describes the Speex header.

8.6 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
- #define JITTER_BUFFER_SET_MARGIN 0
- #define JITTER_BUFFER_GET_MARGIN 1
- #define JITTER_BUFFER_GET_AVALIABLE_COUNT 3
- #define **JITTER_BUFFER_ADJUST_INTERPOLATE** -1
- #define **JITTER_BUFFER_ADJUST_OK** 0
- #define **JITTER_BUFFER_ADJUST_DROP** 1

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_int32_t ∗start_offset)
- int jitter_buffer_get_pointer_timestamp (JitterBuffer ∗jitter)
- void jitter_buffer_tick (JitterBuffer ∗jitter)
- int jitter_buffer_ctl (JitterBuffer ∗jitter, int request, void ∗ptr)
- int **jitter_buffer_update_delay** (JitterBuffer ∗jitter, JitterBufferPacket ∗packet, spx_int32_- t ∗start_offset)
- 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)

8.6.1 Detailed Description

Adaptive jitter buffer for Speex.

8.7 speex_preprocess.h File Reference

Speex preprocessor. The preprocess can do noise suppression, residual echo suppression (after using the echo canceller), automatic gain control (AGC) and voice activity detection (VAD).

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

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
- #define SPEEX_PREPROCESS_SET_NOISE_SUPPRESS 18
- #define SPEEX_PREPROCESS_GET_NOISE_SUPPRESS 19
- #define SPEEX_PREPROCESS_SET_ECHO_SUPPRESS 20
- #define SPEEX_PREPROCESS_GET_ECHO_SUPPRESS 21
- #define SPEEX_PREPROCESS_SET_ECHO_SUPPRESS_ACTIVE 22
- #define SPEEX_PREPROCESS_GET_ECHO_SUPPRESS_ACTIVE 23
- #define SPEEX_PREPROCESS_SET_ECHO_STATE 24
- #define SPEEX_PREPROCESS_GET_ECHO_STATE 25
- #define SPEEX_PREPROCESS_SET_AGC_INCREMENT 26
- #define SPEEX_PREPROCESS_GET_AGC_INCREMENT 27
- #define SPEEX_PREPROCESS_SET_AGC_DECREMENT 28
- #define SPEEX_PREPROCESS_GET_AGC_DECREMENT 29
- #define SPEEX_PREPROCESS_SET_AGC_MAX_GAIN 30
- #define SPEEX_PREPROCESS_GET_AGC_MAX_GAIN 31

Typedefs

- typedef SpeexPreprocessState_ SpeexPreprocessState

Functions

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

8.7.1 Detailed Description

Speex preprocessor. The preprocess can do noise suppression, residual echo suppression (after using the echo canceller), automatic gain control (AGC) and voice activity detection (VAD).

8.8 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)

8.8.1 Detailed Description

Describes the handling for intensity stereo.

8.9 speex_types.h File Reference

Speex types.

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

8.9.1 Detailed Description

Speex types.

Index