NAME

libsox_ng – another audio file format and effect library

SYNOPSIS

#include <sox_ng.h>

cc -o file file.c -lsox_ng

DESCRIPTION

libsox_ng is a library of sound sample file format readers/writers and sound effects processors. It is mainly developed to be used by SoX but any audio application might find it useful.

If you want to dive straight in reading and writing audio files, look at the start of **GENERIC FUNCTIONS** and the section on **READING AND WRITING AUDIO FILES**.

#include <sox_ng.h>

Declarations for all of the below (and a lot more!). However, you should only use items whose names begin with **sox_** or **SOX_**; the **lsx_** and **LSX_** items are intended for internal use by SoX's format and effect handlers. They are exported by **libsox** so that dynamically-loaded format handlers and custom format and effect handlers can use them but should not be used by libSoX-based applications.

Programs should normally include **<sox.h>** to be portable to systems that have not upgraded to **sox_ng** yet; those that have upgraded should provide **<sox.h>** as an alias for **<sox_ng.h>** and the interface is unlikely to change.

cc −o *file file.c* **−lsox_ng**

Ensure that if a program includes **<sox.h>**, it is linked with **-lsox**, and **<sox_ng.h>** with **-lsox_ng** as some software distributions (or you) may choose to install them both side by side and the contents of their internal data structures differ.

BASIC TYPES

sox_int8_t	Signed twos-complement 8-bit type
sox_uint8_t	Unsigned 8-bit type
sox_int16_t	Signed twos-complement 16-bit type
sox_uint16_t	Unsigned 16-bit type
sox_int32_t	Signed twos-complement 32-bit type
sox_uint32_t	Unsigned 32-bit type
sox_int64_t	Signed twos-complement 64-bit type
sox_uint64_t	Unsigned 64-bit type
sox_int24_t	Alias for **sox_int32_t** (beware of the extra byte)
sox_uint24_t	Alias for **sox_uint32_t** (beware of the extra byte)
sox_sample_t	Native SoX audio sample type (alias for **sox_int32_t**)
sox_rate_t	Samples per second are stored as a double
sox_bool	Boolean type: **sox_false** (= 0) or **sox_true** (= 1)

ERROR HANDLING

Most of the functions in libSoX return an integer error code which is **SOX_SUCCESS** if the functions succeeded or a **sox_error_t** which can take the following values:

SOX_SUCCESS	Function succeeded = 0
SOX_EOF	End Of File or other error
SOX_EHDR	Invalid Audio Header
SOX_EFMT	Unsupported data format
SOX_ENOMEM	Can't alloc memory
SOX_EPERM	Operation not permitted
SOX_ENOTSUP	Operation not supported
SOX_EINVAL	Invalid argument
SOX_ENOKEYMAP	No such keymap
SOX_ENOEFFECT	No such effect

or other values mapped from **errno**(3).

char *sox_strerror(int *sox_errno***)**
Converts a SoX error code into an error string and returns the error string corresponding to the specified error code, or a generic message if the error code is not recognized.

GENERIC FUNCTIONS

int sox_init(void)
Initialize the effects library. Returns **SOX_SUCCESS** if successful.

int sox_format_init(void)
Find and load format handler plugins. Returns **SOX_SUCCESS** if successful.

void sox_format_quit(void)
Unload format handler plugins.

int sox_quit(void)
Close the effects library and unload format handler plugins. Returns **SOX_SUCCESS** if successful.

char *sox_version(void)
Returns the version number string of libSoX, for example, "14.4.0".

sox_version_info_t *sox_version_info(void)
Returns information about this build of libSoX, containing:

size_t size
Structure size = **sizeof(sox_version_info_t)**

sox_version_flags_t flags
Feature flags bits indicating whether optional features are present in this build of libSoX, the logical OR of:

sox_version_none
No special features (= 0).

sox_version_have_popen
Pipes are available to fetch URLs with **wget** or **curl**, to use external codec programs like **ffmpeg** and for piped inputs like **"|**`sox_ng -n -p synth 1`**"**.

sox_version_have_magic
libmagic is in use to autodetect the formats of files.

sox_version_have_threads
OpenMP support is included, so effects and channels will run in parallel if **sox_get_globals()->use_threads** has not been set to **sox_false**.

sox_version_have_memopen
sox_open_mem_read(), **sox_open_mem_write()** and **sox_open_memstream_write()** do what it says on the box. If **fmemopen**(3) is not available, they fall back to **sox_open_read()** and **sox_open_write()**.

sox_uint32_t version_code
Version number, for example 0x0E0402.

char *version
Version string, for example, "14.4.2".

char *version_extra
Version extra info or **NULL**, set by **PACKAGE_EXTRA**, for example, "beta".

char *distro
Distro or **NULL**, set by **DISTRO**, for example, "Debian".

char *compiler
Compiler info or **NULL**, for example, "msvc 160040219".

char *arch
Architecture-dependent feature code. For example, "1248 48 44 L OMP" where the first four digits give the sizes in bytes of **char**, **short**, **long** and **off_t**, the next two the sizes of **float** and **double**, the next two the sizes of a data pointer and a function pointer, followed by `L` if the machine is little-endian or `B` if big-endian, and `OMP` if multi-threading is available using OpenMP.

sox_globals_t *sox_get_globals(void)
Returns a pointer to the structure with global settings for formats and effects, containing:

char *myname
The name the program was invoked as, for error reporting. **sox_ng** sets this to the bare program name without a directory path (or a `.exe` extension on Windows) as "sox_ng", "soxi_ng", "play_ng" or "rec_ng", or "sox", "soxi", "play" or "rec" when it replaces **sox**.

Other programs using **libsox_ng** can set it to **argv[0]** on startup if they wish, but it is only used by libSoX when creating error messages for **lsx_fail()** when there are syntax errors in effects' options and leaving it **NULL** is harmless.

unsigned verbosity
Messages are output by the default output message handler if **sox_get_globals()->verbosity** >= *level*.

sox_output_message_handler_t output_message_handler
The address of a client-specified message output handling function of type **void** *message_handler*(unsigned *level*, char **filename*, char **fmt*, va_list *ap*)

level 1=FAIL, 2=WARN, 3=INFO, 4=DEBUG, 5=DEBUG_MORE, 6=DEBUG_MOST.

filename Source code `__FILENAME__` from which a message originates.

fmt Message format string.

ap Message format parameters.

For examples of how to use it, see `src/example3.c` and `demo/soxcopy.c`.

sox_bool repeatable
Use pre-determined timestamps and random number generator seed.

size_t bufsiz
Default size (in bytes) used by libSoX for blocks of sample data. Plugins should use similarly-sized buffers to get best performance.

size_t input_bufsiz
Default size (in bytes) used by libSoX for blocks of input sample data. Plugins should use similarly-sized buffers to get best performance.

sox_int32_t ranqd1
Can be used to re-seed libSoX's random number generator.

size_t log2_dft_min_size
Log to base 2 of the minimum size used by libSoX for DFT filtering. Plugins should use similarly-sized DFTs to get best performance.

float A4
The frequency in Hz of A above middle C, usually 440, that is used when converting note names to frequencies.

FILE FORMAT HANDLERS

Format handlers deal with decoding and encoding audio files and are accessed with the following types and functions:

sox_format_handler_t
Handler structure defined by each format, containing:

unsigned sox_lib_version_code
Checked when loading format handler plugins.

char *description
A short description of the format.

char **names
Null-terminated array of filename extensions handled by this format.

unsigned int flags
File flags, the logical OR of:

SOX_FILE_NOSTDIO	Does not use stdio routines
SOX_FILE_DEVICE	File is an audio device
SOX_FILE_PHONY	Phony file/device (for example /dev/null)
SOX_FILE_REWIND	File should be rewound to write header
SOX_FILE_BIT_REV	Is the file bit-reversed?
SOX_FILE_NIB_REV	Is the file nibble-reversed?
SOX_FILE_ENDIAN	Is the file format endian?
SOX_FILE_ENDBIG	For an endian file format, is it big endian?
SOX_FILE_MONO	Do channel restrictions allow mono?
SOX_FILE_STEREO	Do channel restrictions allow stereo?
SOX_FILE_QUAD	Do channel restrictions allow quad?
SOX_FILE_LIT_END	A mask to OR in if the file is little-endian
SOX_FILE_BIG_END	A mask to OR in if the file is big-endian
SOX_FILE_CHANS	A mask to interrogate channels restrictions. If **flags & SOX_FILE_CHANS** is 0, there are no restrictions.

int (*startread)(sox_format_t *_ft_)
The function to initialize the decoder. If it is **NULL**, the format cannot be read.

size_t (*read)(sox_format_t *_ft_, sox_sample_t *_buf_, size_t _len_)
Decode a block of samples.

int (*stopread)(sox_format_t *_ft_)
Close reader (decoder); **NULL** if no closing necessary.

int (*startwrite)(sox_format_t *_ft_)
The function to initialize the encoder. If it is **NULL**, the format cannot be written.

size_t (*write)(sox_format_t *_ft_, sox_sample_t *_buf_, size_t _len_);
Encode a block of sample.

int (*stopwrite)(sox_format_t *_ft_)
Close writer (decoder); **NULL** if no closing necessary.

int (*seek)(sox_format_t *_ft_, sox_uint64_t _offset_)
Reposition reader; **NULL** if not supported.

unsigned *write_formats
An array of values indicating the encodings and precisions supported for writing. The data format is encoding, precision, precision, ..., 0, repeat, and end with 0. The default precision is given first. An example for a format supporting signed values at 16 and 24 bits, defaulting to 16 bits, and unsigned at 8 bits:

```
unsigned *formats = {
  SOX_ENCODING_SIGN2, 16, 24, 0,
  SOX_ENCODING_UNSIGNED, 8, 0,
  0
}
```

sox_rate_t *write_rates
A null-terminated array of sample rates supported for writing, **NULL** if all (or almost all) rates are supported.

size_t priv_size
SoX automatically allocates a buffer in which the handler can store data, of this size in bytes.

sox_format_t
Data passed to and from format handlers, containing:

char *filename
The file's name.

sox_signalinfo_t signal
Signal specifications for the reader (decoder) or the writer (encoder): sample rate, number of channels, precision, length, headroom multiplier. Info will be **SOX_UNSPEC** if the user provided no info.

sox_encodinginfo_t encoding
Encoding specifications for the reader or writer.

char *filetype
The type of the file, as determined by inspection of the file's header or by **libmagic**.

sox_oob_t oob
Out-Of-Band data: comments (ID3 tags), instrument info, loop info.

sox_bool seekable
Can seek on this file.

sox_bool last_byte_was_zero
The last byte written was a zero.

char mode
Read or write mode ('r' or 'w').

sox_uint64_t olength
Samples *chans written to file.

sox_uint64_t clips
Incremented if clipping occurs.

int sox_errno
Failure error code.

char sox_errstr[256]
Failure error text.

void *fp
File stream pointer.

sox_uint8_t *pending_buffer
Buffer of unreturned read bytes.

sox_uint8_t *pending_bytes
Bytes read but not returned yet.

size_t pending_count
How many bytes read but not returned.

lsx_io_type io_type
Whether this stream is a file, a pipe or a URL, one of:

lsx_io_file
Stream is a real file.

lsx_io_pipe
Stream is a pipe.

lsx_io_url
Stream is a URL.

sox_uint64_t tell_off
Current offset within file.

sox_uint64_t data_start
Offset at which sound data begins.

sox_format_handler_t handler
The format handler for this file.

void *priv
The format handler's private data area.

sox_encodinginfo_t
How samples are encoded, containing:

sox_encoding_t encoding
The encoding used for samples.

unsigned bits_per_sample
0 if unknown or variable; uncompressed value if lossless; compressed value if lossy.

double compression
Compression factor (where applicable).

sox_option_t reverse_bytes
Should bytes be reversed?

sox_option_t reverse_nibbles
Should nibbles be reversed?

sox_option_t reverse_bits
Should bits be reversed?

sox_bool opposite_endian
If set to true, the format should reverse its default endianness.

If the **reverse_** fields are **sox_option_default** during **sox_open_read()** or **sox_open_write()**, libSoX will set them to either **sox_option_no** or **sox_option_yes** according to the default of the machine or format.

sox_option_t
The type of the **sox_encodinginfo_t.reverse_*** fields, one of:

sox_option_no	Option specified as no (= 0).
sox_option_yes	Option specified as yes.
sox_option_default	Option unspecified, usually implies some kind of auto-detect logic.

sox_format_tab_t *sox_get_format_fns(void)
Returns a null-terminated table of information about the loaded format handlers, of which each entry contains:

char *name
The name of the format handler.

sox_format_handler_t *(*fn)(void)
The function to call to get a pointer to the format handler's data.

sox_format_handler_t *sox_find_format(char ***name*, sox_bool** *ignore_devices***)**
Finds a format handler by name and returns it, or **NULL** if it is not found.

**sox_bool sox_format_supports_encoding(char ****path*, char **filetype*, sox_encodinginfo_t **encoding*)

path Path to file to be examined (required if *filetype* is **NULL**).

filetype A previously-determined file type, or **NULL** to use the extension of *path*.

encoding Encoding for which the format handler should be queried.

Returns true if the format handler for the specified file type supports the specified encoding.

sox_format_handler_t sox_write_handler(char ****path*, char ****filetype***, char *****filetype1***)**
Gets the format handler for a specified file type. Returns the found format handler, or **NULL** if not found.

path Path to file (required if filetype is **NULL**).

filetype Filetype for which handler is needed, or **NULL** to use the extension from *path*.

filetype1 Receives the filetype that was detected. Pass **NULL** if not needed.

ENCODINGS

sox_encoding_t
The format of encoded sample data, one of:

SOX_ENCODING_UNKNOWN	Encoding has not yet been determined
SOX_ENCODING_SIGN2	Signed linear 2's compliment
SOX_ENCODING_UNSIGNED	Unsigned linear: Sound Blaster
SOX_ENCODING_FLOAT	Floating point (binary format)
SOX_ENCODING_FLOAT_TEXT	Floating point (text format)
SOX_ENCODING_FLAC	FLAC compression
SOX_ENCODING_HCOM	Mac FSSD files with Huffman compression
SOX_ENCODING_WAVPACK	WavPack with integer samples
SOX_ENCODING_WAVPACKF	WavPack with float samples
SOX_ENCODING_ULAW	Mu-law signed logs: US telephony
SOX_ENCODING_ALAW	A-law signed logs: non-US telephony
SOX_ENCODING_G721	G.721 4-bit ADPCM
SOX_ENCODING_G723	G.723 3 or 5 bit ADPCM
SOX_ENCODING_CL_ADPCM	Creative Labs 8->2,3,4 bit compressed PCM
SOX_ENCODING_CL_ADPCM16	Creative Labs 16->4 bit compressed PCM
SOX_ENCODING_MS_ADPCM	Microsoft Compressed PCM
SOX_ENCODING_IMA_ADPCM	IMA Compressed PCM
SOX_ENCODING_OKI_ADPCM	Dialogic/OKI Compressed PCM
SOX_ENCODING_DPCM	Differential PCM: Fasttracker 2 (**xi** format)
SOX_ENCODING_DWVW	Delta Width Variable Word
SOX_ENCODING_DWVWN	Delta Width Variable Word N-bit
SOX_ENCODING_GSM	GSM 6.10 33byte frame lossy compression
SOX_ENCODING_MP1	MPEG 1 Layer 1 compression
SOX_ENCODING_MP2	MPEG 1 Layer 2 compression
SOX_ENCODING_MP3	MPEG 1 Layer 3 compression
SOX_ENCODING_VORBIS	Vorbis compression
SOX_ENCODING_AMR_WB	AMR-WB compression
SOX_ENCODING_AMR_NB	AMR-NB compression
SOX_ENCODING_CVSD	Continuously Variable Slope Delta
SOX_ENCODING_LPC10	Linear Predictive Coding
SOX_ENCODING_OPUS	Opus compression
SOX_ENCODING_DSD	Direct Stream Digital
SOX_ENCODINGS	End of list marker

sox_encodings_info_t *sox_get_encodings_info(void)
Returns a pointer to a null-terminated list of available encodings, containing:

sox_encodings_flags_t flags
How lossy the format is, one of:

sox_encodings_none
No flags specified (implies lossless encoding).

sox_encodings_lossy1
Encode, decode: lossy once.

sox_encodings_lossy2
Encode, decode, encode, decode: lossy twice.

char *name
Short name for the encoding.

char *desc
Description of the encoding.

void sox_init_encodinginfo(sox_encodinginfo_t *_e_)
Fills in a **sox_encodinginfo_t** with default values.

unsigned sox_precision(sox_encoding_t *encoding***, unsigned** *bits***)**
Given an encoding and the encoded bits_per_sample, returns the number of useful bits per sample in the decoded data, or 0 to indicate that the value returned by the format handler should be used instead of a predetermined precision.

encoding Encoding for which to lookup precision information.

bits The number of encoded bits per sample.

READING AND WRITING AUDIO FILES

sox_format_t *sox_open_read(char *_path_, sox_signalinfo_t *_signal_, sox_encodinginfo_t *_encoding_, char *_filetype_)
Opens a decoding session for a file.

path Path to file to be opened (required).

signal Information already known about audio stream, or **NULL** if none.

encoding Information already known about sample encoding, or **NULL** if none.

filetype Previously-determined file type, or **NULL** to auto-detect.

Returns the handle for the new session, which must be closed with **sox_close()**, or **NULL** on failure.

sox_format_t *sox_open_mem_read(void *_buffer_, size_t *size***, sox_signalinfo_t *_signal_, sox_encodinginfo_t *_encoding_, char *_filetype_)**
Opens a decoding session for a memory buffer.

buffer Pointer to audio data buffer (required).

size Number of bytes to read from audio data buffer.

signal Information already known about audio stream, or **NULL** if none.

encoding Information already known about sample encoding, or **NULL** if none.

filetype Previously-determined file type, or **NULL** to auto-detect.

Returns a handle for the new session, which must be closed with **sox_close()**, or **NULL** on failure.

size_t sox_read(sox_format_t *_ft_, sox_sample_t *_buf_, size_t *len***)**
Reads samples from a decoding session into a sample buffer.

buf Buffer from which to read samples.

len Number of samples available in buf.

Returns the number of samples decoded, or 0 on EOF or a read error. If it was the end of the file, **errno** will be zero, non-zero if there was a read error.

int sox_seek(sox_format_t *_ft_, sox_uint64_t _offset_, int _whence_)
Sets the location at which the next samples will be decoded.

offset Sample offset at which to position reader.

whence Only **SOX_SEEK_SET** is currently supported.

Returns **SOX_SUCCESS** if successful.

sox_format_t *sox_open_write(char *_path_, sox_signalinfo_t *_signal_, sox_encodinginfo_t *_encoding_, char *_filetype_, sox_oob_t *_oob_, sox_bool (*_overwrite_)(char *_filename_))
Opens an encoding session for a file.

path Path to file to be written (required).

signal Information about desired audio stream (required).

encoding Information about desired sample encoding, or **NULL** to use defaults.

filetype Previously-determined file type, or **NULL** to auto-detect.

oob Out-of-band data to add to file, or **NULL** if none.

overwrite Called if file exists to determine whether overwrite is ok.

Returns the new session handle, which must be closed with **sox_close**(), or **NULL** on failure.

sox_format_t *sox_open_mem_write(void *_buffer_, size_t _size_, sox_signalinfo_t *_signal_, sox_encodinginfo_t *_encoding_, char *_filetype_, sox_oob_t *_oob_)
Opens an encoding session for a memory buffer.

buffer A pointer to the audio data buffer that receives data (required).

size The maximum number of bytes to write to audio data buffer.

signal Information about the desired audio stream (required).

encoding Information about the desired sample encoding, or **NULL** to use defaults.

filetype The previously-determined file type, or **NULL** to auto-detect.

oob Out-of-band data to add to file, or **NULL** if none.

Returns the new session handle, which must be closed with **sox_close**(), or **NULL** on failure.

sox_format_t *sox_open_memstream_write(char **_buffer_p_, size_t *_size_p_, sox_signalinfo_t *_signal_, sox_encodinginfo_t *_encoding_, char *_filetype_, sox_oob_t *_oob_)
Opens an encoding session for a memstream buffer.

buffer_p Receives pointer to audio data buffer that receives data (required).

size_p Receives size of data written to audio data buffer (required).

signal Information about desired audio stream (required).

encoding Information about desired sample encoding, or **NULL** to use defaults.

filetype Previously-determined file type, or **NULL** to auto-detect.

oob Out-of-band data to add to file, or **NULL** if none.

Returns the new session handle, which must be closed with **sox_close**(), or **NULL** on failure.

size_t sox_write(sox_format_t *_ft_, sox_sample_t *_buf_, size_t _len_)
Writes samples to an encoding session from a sample buffer.

buf Buffer from which to read samples.

len Number of samples available in buf.

Returns the number of samples encoded, or zero if a write error occurred (for the reason, check **errno**(3).)

int sox_close(sox_format_t *_ft_)
Closes an encoding or decoding session. Returns **SOX_SUCCESS** if successful.

EFFECTS AND EFFECTS CHAINS

The following types and functions are used to access libSoX's powerful effects.

sox_effect_t
Effect information, containing:

sox_effects_globals_t *global_info
Global effects parameters, containing:

sox_plot_t plot
The type of plot requested, one of:

sox_plot_off
No plot.

sox_plot_octave
Octave plot.

sox_plot_gnuplot
Gnuplot plot.

sox_plot_data
Output the plot data as text.

sox_globals_t *global_info
A pointer to the associated **sox_globals_t**.

sox_signalinfo_t in_signal
Info about the incoming data stream.

sox_signalinfo_t out_signal
Info about the outgoing data stream.

sox_encodinginfo_t *in_encoding
Info about the incoming data encoding.

sox_encodinginfo_t *out_encoding
Info about the outgoing data encoding.

sox_effect_handler_t handler
The handler for this effect.

sox_uint64_t clips
Incremented when clipping occurs.

size_t flows
1 if **handler.flags & SOX_EFF_MCHAN**, or the number of channels otherwise.

size_t flow
Flow number.

void *priv
The effect's private data area (each flow has a separate copy).

sox_signalinfo_t
Signal parameters, containing:

sox_rate_t rate
Samples per second, 0 if unknown.

unsigned channels
Number of sound channels, 0 if unknown.

unsigned precision
Bits per sample, 0 if unknown.

sox_uint64_t length
Samples in the file (sample frames × channels), **SOX_UNSPEC** (= 0) if the actual value is not yet known, **SOX_UNKNOWN_LEN** is used within the effects chain if the actual length is not known. Format handlers use SOX_UNSPEC instead. **SOX_IGNORE_LENGTH** to indicate that a format handler should ignore length information in file headers.

double *mult
Effects headroom multiplier; may be **NULL**.

Members are **SOX_UNSPEC** (= 0) if the actual value is not yet known.

sox_effect_handler_t
Effect handler information, containing:

char *name
The effect's name.

char *usage
One-line effect usage and parameters.

char **extra_usage
Additional lines of usage, null-terminated.

unsigned int flags
The logical OR of:

SOX_EFF_CHAN
The effect might alter the number of channels.

SOX_EFF_RATE
The effect might alter sample rate.

SOX_EFF_PREC
The effect does its own calculation of output sample precision (otherwise a default value is taken, depending on the presence of SOX_EFF_MODIFY).

SOX_EFF_LENGTH
The effect might alter audio length (as measured in time units, not necessarily in samples).

SOX_EFF_MCHAN
The effect handles multiple channels internally.

SOX_EFF_NULL
The effect does nothing and can be optimized out of the chain.

SOX_EFF_GAIN
The effect does not support **gain −r**.

SOX_EFF_MODIFY
The effect does not modify sample values (but it might remove or duplicate samples or insert zeros).

SOX_EFF_INTERNAL
The effect is present in libSoX but is not valid for use by SoX command-line tools.

int (*getopts)(sox_effect_t *effp, int argc, char **argv)
The effect's function to parse command-line arguments, called once per effect.

int (*start)(sox_effect_t *effp)
The function to initialize effect, called once per flow.

int (*flow)(sox_effect_t *effp, sox_sample_t *ibuf, sox_sample_t *obuf, size_t *isamp, size_t *osamp)
The function to process samples.

ibuf Buffer from which to read samples

obuf Buffer to which samples are written.

isamp On entry, contains the capacity of *ibuf* in samples; on exit, contains the number of samples consumed.

osamp On entry, contains the capacity of *obuf* in samples; on exit, contains the number of samples written.

int (*drain)(sox_effect_t *effp, sox_sample_t *obuf, size_t *osamp)
The function to output after input is complete. Parameters are as for **flow**.

int (*stop)(sox_effect_t *effp)
The function to shut the effect down (called once per flow).

int (*kill)(sox_effect_t *effp)
The function to shut the effect down (called once per effect).

char *(*get)(sox_effect_t *effp, char *field)
Pointer to a function to read the current value of an effect's parameter, returning **NULL** if there is no readable parameter of that name, or a pointer to mallocked memory containing a string value sprintfed with "%g" which it is the caller's responsibility to free.

char *(*set)(sox_effect_t *effp, char *field, char *value)
Pointer to a function to set the value of an effect's parameter, returning **NULL** if that parameter cannot be set or if the value is garbage, or a pointer to mallocked memory containing the new value sprintfed with "%g". Values outside the parameter's range set the minimum or maximum value and return that.

size_t priv_size
The size of the effect's private data.

sox_effects_chain_t
A chain of effects to be applied to a stream, containing:

sox_effect_t **effects
Table of effects to be applied to a stream.

size_t length
Number of effects to be applied.

sox_effects_globals_t global_info
Copy of global effects settings.

sox_encodinginfo_t *in_enc
Input encoding.

sox_encodinginfo_t *out_enc
Output encoding.

sox_effects_globals_t *sox_get_effects_globals(void)
Returns global parameters for effects.

sox_effect_handler_t *sox_find_effect(char ****name*)**
Finds the effect handler with the given name. Returns an effect pointer, or **NULL** if not found.

sox_effect_t *sox_create_effect(sox_effect_handler_t ****eh*)**
Creates an effect using the given handler. Returns the new effect, or **NULL** if not found.

int sox_effect_options(sox_effect_t ****effp*, int** *argc***, char *****argv***)**
Applies the command-line options to the effect. Returns the number of arguments consumed.

sox_effect_handler_t *sox_get_effect_fns(void)
Returns an null-terminated array of the known effect handlers.

sox_effects_chain_t *sox_create_effects_chain(sox_encodinginfo_t *_in_enc_, sox_encodinginfo_t *_out_enc_)
Initializes an effects chain.

in_enc Input encoding.

out_enc Output encoding.

Returns a handle, which must be closed with **sox_delete_effects_chain()**. or **NULL** on failure.

void sox_delete_effects_chain(sox_effects_chain_t *_chain_)
Closes an effects chain and deletes all the effects in it.

int sox_add_effect(sox_effects_chain_t *_chain_, sox_effect_t *_effp_, sox_signalinfo_t *_in_, sox_signalinfo_t *_out_)
Adds an effect to the end of the effects chain. Returns **SOX_SUCCESS** if successful.

in Characteristics of the input signal to the chain.

out Characteristics of the output signal from the chain.

int sox_flow_effects(sox_effects_chain_t *_chain_, int (*_callback_)(sox_bool _all_done_, void *_data), void *_data_)
Runs the effects chain. Returns **SOX_SUCCESS** if successful.

chain Effects chain to run.

callback If not **NULL**, a pointer to a function for monitoring flow progress, called between processing each block of data. It is handed a copy of the *client_data* pointer that was passed to **sox_flow_effects** and if the callback returns anything other than **SOX_SUCCESS**, the flow is stopped.

data A pointer that is pass to the callback.

sox_uint64_t sox_effects_clips(sox_effects_chain_t *_chain_)
Returns the number of clips that occurred while running an effects chain.

sox_uint64_t sox_stop_effect(sox_effect_t *_effp_)
Shuts down an effect (calls stop on each of its flows) and returns the number of clips from all flows.

void sox_push_effect_last(sox_effects_chain_t *_chain_, sox_effect_t *_effp_)
Adds an already-initialized effect to the end of the chain.

sox_effect_t *sox_pop_effect_last(sox_effects_chain_t *_chain_)
Removes and returns an effect from the end of the chain. Returns the removed effect, or **NULL** if the chain has no effects.

void sox_delete_effect(sox_effect_t *_effp_)
Shut down and delete an effect.

void sox_delete_effect_last(sox_effects_chain_t *_chain_)
Shut down and delete the last effect in the chain.

void sox_delete_effects(sox_effects_chain_t *_chain_)
Shut down and delete all effects in the chain.

EFFECT-SPECIFIC FUNCTIONS

For both of these, *_effp_ should be a **trim** effect.

sox_uint64_t sox_trim_get_start(sox_effect_t *_effp_)
Gets the sample offset of the start of the **trim**, useful for efficiently skipping the part that will be trimmed anyway (get trim start, seek, then clear trim start). Returns the sample offset of the start of the **trim**.

void sox_trim_clear_start(sox_effect_t *_effp_)
Clears the start of the trim to 0.

OUT-OF-BAND DATA

sox_oob_t

Out-Of-Band data: comments, instrument info and loop info, containing:

sox_comments_t comments

File's metadata as comment strings in id=value format, accessed via the **sox_*_comments** functions below.

sox_instrinfo_t instr

Instrument specification, containing:

signed char MIDInote

For unity pitch playback.

signed char MIDIlow

MIDI pitch-bend low range.

signed char MIDIhi

MIDI pitch-bend high range.

unsigned char loopmode

Loop modes: the lower 4 bits describe the loop behaviour, one of:

sox_loop_none

Single-shot.

sox_loop_forward

Forward loop.

sox_loop_forward_back.

Forward/back loop.

and the upper 4 bits mask the loop blass, the logical OR of

sox_loop_8

8 loops (??).

sox_loop_sustain_decay

AIFF style, one sustain & one decay loop.

unsigned nloops

Number of active loops (max **SOX_MAX_NLOOPS** = 8).

sox_loopinfo_t loops[SOX_MAX_NLOOPS];

Looping specifications, containing:

sox_uint64_t start

First sample.

sox_uint64_t length

Length of loop.

unsigned count

Number of repeats, 0=forever.

unsigned char type

Bits with the same meaning as **loopmode** above.

COMMENTS

size_t sox_num_comments(sox_comments_t *comments***)**

Returns the number of items in the metadata block.

void sox_append_comment(sox_comments_t * *comments***, char *** *item***)**

Adds an item to the metadata block, in "id=value" format.

void sox_append_comments(sox_comments_t *_comments_, char *_items_)
Adds a newline-separated list of "id=value" items to the metadata block, for example "id1=value1\nid2=value2".

sox_comments_t sox_copy_comments(sox_comments_t _comments_)
Duplicates the metadata block and returns the copy.

void sox_delete_comments(sox_comments_t *_comments_)
Frees the metadata block.

char *sox_find_comment(sox_comments_t _comments_, char *_id_)
If "id=value" is found in the comments, returns a pointer to its value, or **NULL** if the id was not found.

PLAYLIST FUNCTIONS

sox_bool sox_is_playlist(char *_filename_)
Returns true if the specified file is a known playlist file type.

int sox_parse_playlist(int (*_callback_)(void *_data_, char *_filename_), void *_data_, char *_listname_)
Parses the specified playlist file. Returns **SOX_SUCCESS** if successful.

callback The function to call for each item in the playlist.

data A pointer to client data, passed to the callback function.

listname The name of the playlist file.

KEYMAPS

Keymaps are the mechanism by which SoX allows you to modify effects' internal parameters on the fly by mapping keys with **−−keymap** and then pressing them while in **−−interactive** mode (e.g. when using **play** or with **−d** as the output filename.)

void sox_keymap_add(char *_key_, char *_effect_, char *_field_, char _op_, double _step_)
Add a keymap.

key The string name of the key to map, for example "D".

effect The name of the effect to affect. "echo" will affect all echoes in the effects chain; "echo2" will only affect the second instance of it.

field The name of the parameter in the effect's priv_t to change. For effects with multiple stages, "decay" will affect the decay of all stages and "decay2" will only affect the decay of the scond stage.

op 10 How to affect the parameter: '+', '−', '*', '/' or '='.

step 10 How much to add or subtract from the parameter, to muliply or divide it by, or to set its value to.

extern sox_bool sox_is_keymapped(char *_key_)
See if a key or an effect.field is used in a keymap

int sox_keymap_apply(sox_effects_chain_t *_effp_, char *_key_)
Apply a keymap. Returns **SOX_SUCCESS** on successful application, **SOX_ENOEFFECT** if the effect was not found in the chain or it was found but doesn't have a keymappable parameter of that name, **SOX_ENOKEYMAP** if the key was not mapped to anything.

void sox_keymap_free(void)
Forget all keymaps.

MISCELLANEOUS UTILITY FUNCTIONS

size_t sox_basename(char *_buffer_, size_t _len_, char *_filename_)
Gets the basename of the specified file; for example, the basename of "/a/b/c.d" would be "c". Returns the number of characters written to the buffer, excluding the final nul character, or 0 on failure.

buffer Buffer into which basename should be written.

len Size of *buffer*, in bytes.

filename Filename from which to extract basename.

GENERIC MACROS

SOX_LIB_VERSION(*a***, *b*, *c*)**
Compute a 32-bit integer API version from three 8-bit parts. *a* is the major version number, *b* the minor version and *c* the micro or bugfix version. Returns a 32-bit integer API version like 0x000a0b0c.

SOX_LIB_VERSION_CODE
The current API version as a 32-bit integer which follows the version number of SoX.

SOX_INT_MIN(*bits***)**
Returns the smallest (most negative) value storable in a twos-complement signed integer with the specified number of bits, cast to an unsigned integer. For example, **SOX_INT_MIN(8)** = 0x80, **SOX_INT_MIN(16)** = 0x8000, etc.

bits The size of value for which to calculate minimum and maximum values.

SOX_INT_MAX(*bits***)**
Returns the largest (positive) value storable in a twos-complement signed integer with the specified number of bits, cast to an unsigned integer for example, **SOX_INT_MAX(8)** = 0x7F, **SOX_INT_MAX(16)** = 0x7FFF, etc.

SOX_UINT_MAX(*bits***)**
Returns the largest value storable in an unsigned integer with the specified number of bits; for example, **SOX_UINT_MAX(8)** = 0xFF, **SOX_UINT_MAX(16)** = 0xFFFF, etc.

SOX_INT8_MAX
Returns 0x7F.

SOX_INT16_MAX
Returns 0x7FFF.

SOX_INT24_MAX
Returns 0x7FFFFF.

SOX_INT32_MAX
Returns 0x7FFFFFFF.

SOX_SAMPLE_PRECISION
Bits in a **sox_sample_t** (= 32).

SOX_SAMPLE_MAX
The maximum value of a **sox_sample_t** (= 0x7FFFFFFF).

SOX_SAMPLE_MIN
The minimum (most negative) value of a **sox_sample_t** (= 0x80000000).

SOX_SAMPLE_NEG
The sign bit for **sox_sample_t** (= 0x80000000).

SOX_SIZE_MAX
The maximum value of a **size_t**.

SAMPLE CONVERSION

Linear PCM <--> sox_sample_t

I/O Format	Input Minimum	sox_sample_t Minimum	Clips I	Clips O	Input Maximum	sox_sample_t Maximum	Clips I	Clips O
Float	−inf	−1	y	n	inf	1 − 5e−10	y	n
Int8	−128	−128	n	n	127	127.9999999	n	y

Int16	−32768	−32768	n	n	32767	32767.99998	n	y
Int24	−8388608	−8388608	n	n	8388607	8388607.996	n	y
Int32	−2147483648	−2147483648	n	n	2147483647	2147483647	n	n

Conversions are as accurate as possible (with rounding).

Halves are rounded toward +infinity, all others to nearest integer.

Clips shows whether on not there is the possibility of a conversion clipping to the minimum or maximum value when inputting from or outputting to a given type.

Unsigned integers are converted to and from signed integers by flipping the uppermost bit then treating them as signed integers.

Before you use the following SoX sample conversion macros in a function, **SOX_SAMPLE_LOCALS** declares the temporary local variables they require. For example:

```
sox_int16_t sox_sample_to_CD(sox_sample_t sample)
{
  SOX_SAMPLE_LOCALS
  unsigned clips = 0;

  return SOX_SAMPLE_TO_SIGNED(16, sample, clips);
}
```

The following macros return a SoX native sample value and parameters are:

bits The width of the resulting sample (1 through 32).

d The value to be converted.

clips A variable that is incremented if the result is too big.

These macros are largely agnostic to the numeric types of their arguments.

SOX_SIGNED_TO_SAMPLE(*bits***,** *d***)**
Converts a signed integer of width *bits* to **sox_sample_t**.

SOX_UNSIGNED_TO_SAMPLE(*bits***,** *d***)**
Converts an unsigned integer of width *bits* to **sox_sample_t**.

SOX_UNSIGNED_8BIT_TO_SAMPLE(sox_uint8_t *d***,** *clips***)**
Converts an unsigned 8-bit integer to **sox_sample_t**. The *clips* parameter is not used.

SOX_SIGNED_8BIT_TO_SAMPLE(sox_int8_t *d***,** *clips***)**
Converts a signed 8-bit integer to **sox_sample_t**. The *clips* parameter is not used.

SOX_UNSIGNED_16BIT_TO_SAMPLE(sox_uint16_t *d***,** *clips***)**
Converts an unsigned 16-bit integer to **sox_sample_t**. The *clips* parameter is not used.

SOX_SIGNED_16BIT_TO_SAMPLE(sox_int16_t *d***,** *clips***)**
Converts a signed 16-bit integer to **sox_sample_t**. The *clips* parameter is not used.

SOX_UNSIGNED_24BIT_TO_SAMPLE(sox_int24_t *d***,** *clips***)**
Converts an unsigned 24-bit integer to **sox_sample_t**. The *clips* parameter is not used.

SOX_SIGNED_24BIT_TO_SAMPLE(sox_int24_t *d***,** *clips***)**
Converts a signed 24-bit integer to **sox_sample_t**. The *clips* parameter is not used.

SOX_UNSIGNED_32BIT_TO_SAMPLE(sox_uint32_t *d***,** *clips***)**
Converts an unsigned 32-bit integer to **sox_sample_t**. The *clips* parameter is not used.

SOX_SIGNED_32BIT_TO_SAMPLE(sox_int32_t *d***,** *clips***)**
Converts a signed 32-bit integer to **sox_sample_t**. The *clips* parameter is not used.

SOX_FLOAT_32BIT_TO_SAMPLE(float *d*, *clips*)
Converts a 32-bit float to **sox_sample_t**.

SOX_FLOAT_64BIT_TO_SAMPLE(double *d*, *clips*)
Converts a 64-bit float to **sox_sample_t**.

SOX_SAMPLE_TO_UNSIGNED(*bits*, **sox_sample_t** *samp*, *clips*)
Converts a SoX native sample to an unsigned integer of width *bits*.

SOX_SAMPLE_TO_SIGNED(*bits*, **sox_sample_t** *samp*, *clips*)
Converts a SoX native sample to a signed integer of width *bits*.

SOX_SAMPLE_TO_UNSIGNED_8BIT(sox_sample_t *samp*, *clips*)
Converts a SoX native sample to an unsigned 8-bit integer.

SOX_SAMPLE_TO_SIGNED_8BIT(sox_sample_t *samp*, *clips*)
Converts a SoX native sample to an signed 8-bit integer.

SOX_SAMPLE_TO_UNSIGNED_16BIT(sox_sample_t *samp*, *clips*)
Converts a SoX native sample to an unsigned 16-bit integer.

SOX_SAMPLE_TO_SIGNED_16BIT(sox_sample_t *samp*, *clips*)
Converts a SoX native sample to a signed 16-bit integer.

SOX_SAMPLE_TO_UNSIGNED_24BIT(sox_sample_t *samp*, *clips*)
Converts a SoX native sample to an unsigned 24-bit integer.

SOX_SAMPLE_TO_SIGNED_24BIT(sox_sample_t *samp*, *clips*)
Converts a SoX native sample to a signed 24-bit integer.

SOX_SAMPLE_TO_UNSIGNED_32BIT(sox_sample_t *samp*, *clips*)
Converts a SoX native sample to an unsigned 32-bit integer. The *clips* parameter is not used.

SOX_SAMPLE_TO_SIGNED_32BIT(sox_sample_t *samp*, *clips*)
Converts a SoX native sample to a signed 32-bit integer. The *clips* parameter is not used.

SOX_SAMPLE_TO_FLOAT_32BIT(sox_sample_t *samp*, *clips*)
Converts a SoX native sample to a 32-bit float.

SOX_SAMPLE_TO_FLOAT_64BIT(sox_sample_t *samp*, *clips*)
Converts a SoX native sample to a 64-bit float. The *clips* parameter is not used.

SOX_SAMPLE_CLIP_COUNT(*d*, *clips*)
Clips a value of a type that is larger then **sox_sample_t** (for example, **sox_int64_t**) to **sox_sample_t**'s limits and increment a counter if clipping occurs. *d* is the value (an lvalue) to be clipped, updated as necessary.

SOX_ROUND_CLIP_COUNT(*d*, *clips*)
Clips a value of a type that is larger then **sox_sample_t** (for example, **sox_int64_t**) to **sox_sample_t**'s limits and increment a counter if clipping occurs. Returns the clipped value.

SOX_INTEGER_CLIP_COUNT(*bits*, *d*, *clips*)
Clips a value to the limits of a signed integer of the specified width and increment a counter if clipping occurs. Returns the clipped value.

SOX_16BIT_CLIP_COUNT(*d*, *clips*)
Clips a value to the limits of a 16-bit signed integer and increment a counter if clipping occurs. Returns the clipped value.

SOX_24BIT_CLIP_COUNT(*d*, *clips*)
Clips a value to the limits of a 24-bit signed integer and increments a counter if clipping occurs. Returns the clipped value.

SOX_DEFAULT_CHANNELS
The default channel count is 2 (stereo).

SOX_DEFAULT_RATE
The default sample rate is 48000Hz.

SOX_DEFAULT_PRECISION
The default precision is 16 bits per sample.

SOX_DEFAULT_ENCODING
The default encoding is **SOX_ENCODING_SIGN2** (linear 2's complement PCM).

LINKING

How you link against libsox_ng depends on how SoX was built on your system. For a static build, just link against the library. For a dynamic build, use **libtool** to link with the correct linker flags. See the **libtool** manual for details; basically, you use it like this:

```
libtool --mode=link gcc -o prog /path/to/libsox_ng.la
```

COPYRIGHT

Copyright 1991−2015 Lance Norskog, Chris Bagwell and sundry contributors.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

AUTHORS

The many authors and contributors are listed in the README file that is distributed with the source code.

SEE ALSO

sox_ng(1), **libsoxeffect_ng**(3), **soxformat_ng**(7), `src/example*.c` and `demo/*.c` in the SoX source distribution.