

**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**

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

<b>SOX_SUCCESS</b>	Function succeeded = 0
<b>SOX_EOF</b>	End Of File or other error
<b>SOX_EHDR</b>	Invalid Audio Header
<b>SOX_EFMT</b>	Unsupported data format
<b>SOX_ENOMEM</b>	Can't alloc memory
<b>SOX_EPERM</b>	Operation not permitted
<b>SOX_ENOTSUP</b>	Operation not supported
<b>SOX_EINVAL</b>	Invalid argument
<b>SOX_ENOKEYMAP</b>	No such keymap
<b>SOX_ENOEFFECT</b>	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**  $\geq$  *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:

<b>SOX_FILE_NOSTDIO</b>	Does not use stdio routines
<b>SOX_FILE_DEVICE</b>	File is an audio device
<b>SOX_FILE_PHONY</b>	Phony file/device (for example /dev/null)
<b>SOX_FILE_REWIND</b>	File should be rewound to write header
<b>SOX_FILE_BIT_REV</b>	Is the file bit-reversed?
<b>SOX_FILE_NIB_REV</b>	Is the file nibble-reversed?
<b>SOX_FILE_ENDIAN</b>	Is the file format endian?
<b>SOX_FILE_ENDBIG</b>	For an endian file format, is it big endian?
<b>SOX_FILE_MONO</b>	Do channel restrictions allow mono?
<b>SOX_FILE_STEREO</b>	Do channel restrictions allow stereo?
<b>SOX_FILE_QUAD</b>	Do channel restrictions allow quad?
<b>SOX_FILE_LIT_END</b>	A mask to OR in if the file is little-endian
<b>SOX_FILE_BIG_END</b>	A mask to OR in if the file is big-endian
<b>SOX_FILE_CHANS</b>	A mask to interrogate channels restrictions. If <b>flags &amp; SOX_FILE_CHANS</b> 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:

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

*op* How to affect the parameter: '+', '-', '\*', '/' or '='.

*step* How much to add or subtract from the parameter, to multiply 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 two's-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 two's-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 0x7FFFFFFF.

### 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 O	Input Maximum	sox_sample_t Maximum	Clips I 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 or 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 a 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 than **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 than **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.