

libtheora Reference Manual
1.0alpha5

Generated by Doxygen 1.4.0

Sat Aug 20 12:41:44 2005

Contents

1	libtheora Main Page	1
1.1	Introduction	1
2	libtheora Directory Hierarchy	3
2.1	libtheora Directories	3
3	libtheora Data Structure Index	5
3.1	libtheora Data Structures	5
4	libtheora File Index	7
4.1	libtheora File List	7
5	libtheora Directory Documentation	9
5.1	/home/giles/projects/xiph/theora/include/ Directory Reference	9
5.2	/home/giles/projects/xiph/theora/include/theora/ Directory Reference	10
6	libtheora Data Structure Documentation	11
6.1	theora_comment Struct Reference	11
6.2	theora_info Struct Reference	13
6.3	theora_state Struct Reference	15
6.4	yuv_buffer Struct Reference	16
7	libtheora File Documentation	17
7.1	theora.h File Reference	17

Chapter 1

libtheora Main Page

1.1 Introduction

This is the documentation for the libtheora C API. libtheora is the reference implementation for **Theora**, a free video codec. Theora is derived from On2's VP3 codec with improved integration for Ogg multimedia formats by Xiph.Org.

Chapter 2

libtheora Directory Hierarchy

2.1 libtheora Directories

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

include	9
theora	10

Chapter 3

libtheora Data Structure Index

3.1 libtheora Data Structures

Here are the data structures with brief descriptions:

theora_comment (Comment header metadata)	11
theora_info (Theora bitstream info)	13
theora_state (Codec internal state and context)	15
yuv_buffer (A YUV buffer for passing uncompressed frames to and from the codec) .	16

Chapter 4

libtheora File Index

4.1 libtheora File List

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

theora.h (The libtheora C API)	17
---	----

Chapter 5

libtheora Directory Documentation

5.1 include/ Directory Reference

Directories

- directory **theora**

5.2 include/theora/ Directory Reference

Files

- file **theora.h**
The libtheora C API.

Chapter 6

libtheora Data Structure Documentation

6.1 theora_comment Struct Reference

Comment header metadata.

```
#include <theora.h>
```

Data Fields

- **char ** user_comments**
An array of comment string vectors.
- **int * comment_lengths**
An array of corresponding string vector lengths in bytes.
- **int comments**
The total number of comment string vectors.
- **char * vendor**
The vendor string identifying the encoder, null terminated.

6.1.1 Detailed Description

Comment header metadata.

This structure holds the in-stream metadata corresponding to the 'comment' header packet.

Meta data is stored as a series of (tag, value) pairs, in length-encoded string vectors. The first occurrence of the '=' character delimits the tag and value. A particular tag may occur more than once. The character set encoding for the strings is always utf-8, but the tag names are limited to case-insensitive ascii. See the spec for details.

In filling in this structure, **theora_decode_header()**(p.22) will null-terminate the user_comment strings for safety. However, the bitstream format itself treats them as 8-bit clean, and so the length array should be treated as authoritative for their length.

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

- **theora.h**

6.2 theora_info Struct Reference

Theora bitstream info.

```
#include <theora.h>
```

Data Fields

- ogg_uint32_t **width**
- ogg_uint32_t **height**
- ogg_uint32_t **frame_width**
- ogg_uint32_t **frame_height**
- ogg_uint32_t **offset_x**
- ogg_uint32_t **offset_y**
- ogg_uint32_t **fps_numerator**
- ogg_uint32_t **fps_denominator**
- ogg_uint32_t **aspect_numerator**
- ogg_uint32_t **aspect_denominator**
- theora_colorspace **colorspace**
- int **target_bitrate**
- int **quality**
Nominal quality setting, 0-63.

- int **quick_p**
Quick encode/decode.

- unsigned char **version_major**
- unsigned char **version_minor**
- unsigned char **version_subminor**
- void * **codec_setup**
- int **dropframes_p**
- int **keyframe_auto_p**
- ogg_uint32_t **keyframe_frequency**
- ogg_uint32_t **keyframe_frequency_force**
- ogg_uint32_t **keyframe_data_target_bitrate**
- ogg_int32_t **keyframe_auto_threshold**
- ogg_uint32_t **keyframe_mindistance**
- ogg_int32_t **noise_sensitivity**
- ogg_int32_t **sharpness**
- theora_pixelformat **pixelformat**

6.2.1 Detailed Description

Theora bitstream info.

Contains the basic playback parameters for a stream, corresponds to the initial 'info' header packet.

Encoded theora frames must be a multiple of 16 is size; this is what the width and height members represent. To handle other sizes, a crop rectangle is specified in frame_height and frame_width, offset_x and offset_y. The offset and size should still be a power of 2 to avoid chroma sampling shifts.

Frame rate, in frames per second is stored as a rational fraction. So is the aspect ratio. Note that this refers to the aspect ratio of the frame pixels, not of the overall frame itself.

see the example code for use of the other parameters and good default settings for the encoder parameters.

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

- **theora.h**

6.3 theora_state Struct Reference

Codec internal state and context.

```
#include <theora.h>
```

Data Fields

- **theora_info** * i
- ogg_int64_t **granulepos**
- void * **internal_encode**
- void * **internal_decode**

6.3.1 Detailed Description

Codec internal state and context.

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

- **theora.h**

6.4 yuv_buffer Struct Reference

A YUV buffer for passing uncompressed frames to and from the codec.

```
#include <theora.h>
```

Data Fields

- **int y_width**
Width of the Y' luminance plane.
- **int y_height**
Height of the luminance plane.
- **int y_stride**
Offset in bytes between successive rows.
- **int uv_width**
Height of the Cb and Cr chroma planes.
- **int uv_height**
Width of the chroma planes.
- **int uv_stride**
Offset between successive chroma rows.
- **unsigned char * y**
Pointer to start of luminance data.
- **unsigned char * u**
Pointer to start of Cb data.
- **unsigned char * v**
Pointer to start of Cr data.

6.4.1 Detailed Description

A YUV buffer for passing uncompressed frames to and from the codec.

This holds a Y'CbCr frame in planar format. The CbCr planes can be subsampled and have their own separate dimensions and row stride offsets. Note that the strides may be negative in some configurations. For theora the width and height of the largest plane must be a multiple of 16. The actual meaningful picture size and offset are stored in the **theora_info**(p.13) structure; frames returned by the decoder may need to be cropped for display. All samples are 8 bits.

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

- **theora.h**

Chapter 7

libtheora File Documentation

7.1 theora.h File Reference

The libtheora C API.

```
#include <ogg/ogg.h>
```

Defines

- `#define OC_FAULT -1`
General failure.
- `#define OC_EINVAL -10`
Library encountered invalid internal data.
- `#define OC_DISABLED -11`
Requested action is disabled.
- `#define OC_BADHEADER -20`
Header packet was corrupt/invalid.
- `#define OC_NOTFORMAT -21`
Packet is not a theora packet.
- `#define OC_VERSION -22`
Bitstream version is not handled.
- `#define OC_IMPL -23`
Feature or action not implemented.
- `#define OC_BADPACKET -24`
Packet is corrupt.
- `#define OC_NEWPACKET -25`
Packet is an (ignorable) unhandled extension.

Typedefs

- typedef **theora_comment** **theora_comment**

Comment header metadata.

Enumerations

- enum **theora_colorspace** { **OC_CS_UNSPECIFIED**, **OC_CS_ITU_REC_470M**, **OC_CS_ITU_REC_470BG**, **OC_CS_NSACES** }

A Colorspace.

- enum **theora_pixelformat** { **OC_PF_420**, **OC_PF_RSVD**, **OC_PF_422**, **OC_PF_444** }

A Chroma subsampling.

Functions

- const char * **theora_version_string** (void)

Retrieve a human-readable string to identify the encoder vendor and version.

- ogg_uint32_t **theora_version_number** (void)

Retrieve a 32-bit version number.

- int **theora_encode_init** (**theora_state** *th, **theora_info** *ti)

Initialize the theora encoder.

- int **theora_encode_YUVin** (**theora_state** *t, **yuv_buffer** *yuv)

Submit a YUV buffer to the theora encoder.

- int **theora_encode_packetout** (**theora_state** *t, int last_p, ogg_packet *op)

Request the next packet of encoded video.

- int **theora_encode_header** (**theora_state** *t, ogg_packet *op)

Request a packet containing the initial header.

- int **theora_encode_comment** (**theora_comment** *tc, ogg_packet *op)

Request a comment header packet from provided metadata.

- int **theora_encode_tables** (**theora_state** *t, ogg_packet *op)

Request a packet containing the codebook tables for the stream.

- int **theora_decode_header** (**theora_info** *ci, **theora_comment** *cc, ogg_packet *op)

Decode an Ogg packet, with the expectation that the packet contains an initial header, comment data or codebook tables.

- int **theora_decode_init** (**theora_state** *th, **theora_info** *c)

*Initialize a **theora_state**(p. 15) handle for decoding.*

- int **theora_decode_packetin** (**theora_state** *th, **ogg_packet** *op)
Input a packet containing encoded data into the theora decoder.
- int **theora_decode_YUVout** (**theora_state** *th, **yuv_buffer** *yuv)
Output the next available frame of decoded YUV data.
- int **theora_packet_isheader** (**ogg_packet** *op)
*Report whether a theora packet is a header or not This function does no verification beyond checking the header flag bit so it should not be used for bitstream identification; use **theora_decode_header**(p.22) for that.*
- int **theora_packet_iskeyframe** (**ogg_packet** *op)
Report whether a theora packet is a keyframe or not.
- int **theora_granule_shift** (**theora_info** *ti)
Report the granulepos shift radix.
- **ogg_int64_t** **theora_granule_frame** (**theora_state** *th, **ogg_int64_t** granulepos)
Convert a granulepos to an absolute frame number.
- double **theora_granule_time** (**theora_state** *th, **ogg_int64_t** granulepos)
Convert a granulepos to absolute time in seconds.
- void **theora_info_init** (**theora_info** *c)
*Initialize a **theora_info**(p.13) structure.*
- void **theora_info_clear** (**theora_info** *c)
*Clear a **theora_info**(p.13) structure.*
- void **theora_clear** (**theora_state** *t)
*Free all internal data associated with a **theora_state**(p.15) handle.*
- void **theora_comment_init** (**theora_comment** *tc)
*Initialize an allocated **theora_comment**(p.11) structure.*
- void **theora_comment_add** (**theora_comment** *tc, char *comment)
*Add a comment to an initialized **theora_comment**(p.11) structure.*
- void **theora_comment_add_tag** (**theora_comment** *tc, char *tag, char *value)
*Add a comment to an initialized **theora_comment**(p.11) structure.*
- char * **theora_comment_query** (**theora_comment** *tc, char *tag, int count)
Look up a comment value by tag.
- int **theora_comment_query_count** (**theora_comment** *tc, char *tag)
Look up the number of instances of a tag.
- void **theora_comment_clear** (**theora_comment** *tc)
*Clear an allocated **theora_comment**(p.11) struct so that it can be freed.*

7.1.1 Detailed Description

The libtheora C API.

7.1.2 Typedef Documentation

7.1.2.1 typedef struct theora_comment theora_comment

Comment header metadata.

This structure holds the in-stream metadata corresponding to the 'comment' header packet.

Meta data is stored as a series of (tag, value) pairs, in length-encoded string vectors. The first occurrence of the '=' character delimits the tag and value. A particular tag may occur more than once. The character set encoding for the strings is always utf-8, but the tag names are limited to case-insensitive ascii. See the spec for details.

In filling in this structure, `theora_decode_header()`(p.22) will null-terminate the user_comment strings for safety. However, the bitstream format itself treats them as 8-bit clean, and so the length array should be treated as authoritative for their length.

7.1.3 Enumeration Type Documentation

7.1.3.1 enum theora_colorspace

A Colorspace.

Enumeration values:

OC_CS_UNSPECIFIED The colorspace is unknown or unspecified.

OC_CS_ITU_REC_470M This is the best option for 'NTSC' content.

OC_CS_ITU_REC_470BG This is the best option for 'PAL' content.

OC_CS_NSACES This marks the end of the defined colorspace.

7.1.3.2 enum theora_pixelformat

A Chroma subsampling.

These enumerate the available chroma subsampling options supported by the theora format. See Section 4.4 of the specification for exact definitions.

Enumeration values:

OC_PF_420 Chroma subsampling by 2 in each direction (4:2:0).

OC_PF_RSVD Reserved value.

OC_PF_422 Horizontal chroma subsampling by 2 (4:2:2).

OC_PF_444 No chroma subsampling at all (4:4:4).

7.1.4 Function Documentation

7.1.4.1 void theora_clear (theora_state * t)

Free all internal data associated with a `theora_state`(p. 15) handle.

Parameters:

t A `theora_state`(p. 15) handle.

7.1.4.2 void theora_comment_add (theora_comment * tc, char * comment)

Add a comment to an initialized `theora_comment`(p. 11) structure.

Parameters:

tc A previously initialized theora comment structure

comment A null-terminated string encoding the comment in the form "TAG=the value"

Neither `theora_comment_add`(p. 21) nor `theora_comment_add_tag`(p. 21) support comments containing null values, although the bitstream format supports this. To add such comments you will need to manipulate the `theora_comment`(p. 11) structure directly.

7.1.4.3 void theora_comment_add_tag (theora_comment * tc, char * tag, char * value)

Add a comment to an initialized `theora_comment`(p. 11) structure.

Parameters:

tc A previously initialized theora comment structure

tag A null-terminated string containing the tag associated with the comment.

value The corresponding value as a null-terminated string

Neither `theora_comment_add`(p. 21) nor `theora_comment_add_tag`(p. 21) support comments containing null values, although the bitstream format supports this. To add such comments you will need to manipulate the `theora_comment`(p. 11) structure directly.

7.1.4.4 void theora_comment_clear (theora_comment * tc)

Clear an allocated `theora_comment`(p. 11) struct so that it can be freed.

Parameters:

tc An allocated `theora_comment`(p. 11) structure.

7.1.4.5 void theora_comment_init (theora_comment * tc)

Initialize an allocated `theora_comment`(p. 11) structure.

Parameters:

tc An allocated `theora_comment`(p. 11) structure

7.1.4.6 `char* theora_comment_query (theora_comment * tc, char * tag, int count)`

Look up a comment value by tag.

Parameters:

tc An initialized `theora_comment`(p. 11) structure

tag The tag to look up

count The instance of the tag. The same tag can appear multiple times, each with a distinct and ordered value, so an index is required to retrieve them all.

Returns:

A pointer to the queried tag's value

Return values:

NULL No matching tag is found

Note:

Use `theora_comment_query_count`(p. 22) to get the legal range for the count parameter.

7.1.4.7 `int theora_comment_query_count (theora_comment * tc, char * tag)`

Look up the number of instances of a tag.

Parameters:

tc An initialized `theora_comment`(p. 11) structure

tag The tag to look up

Returns:

The number on instances of a particular tag.

Call this first when querying for a specific tag and then iterate over the number of instances with separate calls to `theora_comment_query`(p. 22) to retrieve all instances in order.

7.1.4.8 `int theora_decode_header (theora_info * ci, theora_comment * cc, ogg_packet * op)`

Decode an Ogg packet, with the expectation that the packet contains an initial header, comment data or codebook tables.

Parameters:

ci A `theora_info`(p. 13) structure to fill. This must have been previously initialized with `theora_info_init`(p. 27). If *op* contains an initial header, `theora_decode_header`(p. 22) will fill *ci* with the parsed header values. If *op* contains codebook tables, `theora_decode_header`(p. 22) will parse these and attach an internal representation to *ci*->*codec_setup*.

cc A `theora_comment`(p. 11) structure to fill. If *op* contains comment data, `theora_decode_header`(p. 22) will fill *cc* with the parsed comments.

op An `ogg_packet` structure which you expect contains an initial header, comment data or codebook tables.

Return values:

OC_BADHEADER *op* is NULL; OR the first byte of *op->packet* has the signature of an initial packet, but *op* is not a `b_o_s` packet; OR this packet has the signature of an initial header packet, but an initial header packet has already been seen; OR this packet has the signature of a comment packet, but the initial header has not yet been seen; OR this packet has the signature of a comment packet, but contains invalid data; OR this packet has the signature of codebook tables, but the initial header or comments have not yet been seen; OR this packet has the signature of codebook tables, but contains invalid data; OR the stream being decoded has a compatible version but this packet does not have the signature of a theora initial header, comments, or codebook packet

OC_VERSION The packet data of *op* is an initial header with a version which is incompatible with this version of libtheora.

OC_NEWPACKET the stream being decoded has an incompatible (future) version and contains an unknown signature.

0 Success

Note:

The normal usage is that `theora_decode_header()`(p.22) be called on the first three packets of a theora logical bitstream in succession.

7.1.4.9 int theora_decode_init (theora_state * th, theora_info * c)

Initialize a `theora_state`(p. 15) handle for decoding.

Parameters:

th The `theora_state`(p. 15) handle to initialize.

c A `theora_info`(p. 13) struct filled with the desired decoding parameters. This is of course usually obtained from a previous call to `theora_decode_header()`(p. 22).

Return values:

0 Success

7.1.4.10 int theora_decode_packetin (theora_state * th, ogg_packet * op)

Input a packet containing encoded data into the theora decoder.

Parameters:

th A `theora_state`(p. 15) handle previously initialized for decoding.

op An `ogg_packet` containing encoded theora data.

Return values:

0 Success

OC_BADPACKET *op* does not contain encoded video data

7.1.4.11 `int theora_decode_YUVout (theora_state * th, yuv_buffer * yuv)`

Output the next available frame of decoded YUV data.

Parameters:

th A `theora_state`(p. 15) handle previously initialized for decoding.

yuv A `yuv_buffer`(p. 16) in which libtheora should place the decoded data.

Return values:

0 Success

7.1.4.12 `int theora_encode_comment (theora_comment * tc, ogg_packet * op)`

Request a comment header packet from provided metadata.

A pointer to the comment data is placed in a user-provided `ogg_packet` structure.

Parameters:

tc A `theora_comment`(p. 11) structure filled with the desired metadata

op An `ogg_packet` structure to fill. libtheora will set all elements of this structure, including a pointer to the encoded comment data. The memory for the comment data is owned by libtheora.

Return values:

0 Success

7.1.4.13 `int theora_encode_header (theora_state * t, ogg_packet * op)`

Request a packet containing the initial header.

A pointer to the header data is placed in a user-provided `ogg_packet` structure.

Parameters:

t A `theora_state`(p. 15) handle previously initialized for encoding.

op An `ogg_packet` structure to fill. libtheora will set all elements of this structure, including a pointer to the header data. The memory for the header data is owned by libtheora.

Return values:

0 Success

7.1.4.14 `int theora_encode_init (theora_state * th, theora_info * ti)`

Initialize the theora encoder.

Parameters:

th The `theora_state`(p. 15) handle to initialize for encoding.

ti A `theora_info`(p. 13) struct filled with the desired encoding parameters.

Return values:

0 Success

7.1.4.15 `int theora_encode_packetout (theora_state * t, int last_p, ogg_packet * op)`

Request the next packet of encoded video.

The encoded data is placed in a user-provided ogg_packet structure.

Parameters:

t A `theora_state`(p. 15) handle previously initialized for encoding.

last_p whether this is the last packet the encoder should produce.

op An ogg_packet structure to fill. libtheora will set all elements of this structure, including a pointer to encoded data. The memory for the encoded data is owned by libtheora.

Return values:

0 No internal storage exists OR no packet is ready

-1 The encoding process has completed

1 Success

7.1.4.16 `int theora_encode_tables (theora_state * t, ogg_packet * op)`

Request a packet containing the codebook tables for the stream.

A pointer to the codebook data is placed in a user-provided ogg_packet structure.

Parameters:

t A `theora_state`(p. 15) handle previously initialized for encoding.

op An ogg_packet structure to fill. libtheora will set all elements of this structure, including a pointer to the codebook data. The memory for the header data is owned by libtheora.

Return values:

0 Success

7.1.4.17 `int theora_encode_YUVin (theora_state * t, yuv_buffer * yuv)`

Submit a YUV buffer to the theora encoder.

Parameters:

t A `theora_state`(p. 15) handle previously initialized for encoding.

yuv A buffer of YUV data to encode.

Return values:

OC_EINVAL Encoder is not ready, or is finished.

-1 The size of the given frame differs from those previously input

0 Success

7.1.4.18 `ogg_int64_t theora_granule_frame (theora_state * th, ogg_int64_t granulepos)`

Convert a granulepos to an absolute frame number.

The granulepos is interpreted in the context of a given `theora_state`(p. 15) handle.

Parameters:

- th* A previously initialized `theora_state`(p. 15) handle (encode or decode)
- granulepos* The granulepos to convert.

Returns:

The frame number corresponding to *granulepos*.

Return values:

- 1** The given granulepos is undefined (i.e. negative)

This function was added in the 1.0alpha4 release.

7.1.4.19 `int theora_granule_shift (theora_info * ti)`

Report the granulepos shift radix.

When embedded in Ogg, Theora uses a two-part granulepos, splitting the 64-bit field into two pieces. The more-significant section represents the frame count at the last keyframe, and the less-significant section represents the count of frames since the last keyframe. In this way the overall field is still non-decreasing with time, but usefully encodes a pointer to the last keyframe, which is necessary for correctly restarting decode after a seek.

This function reports the number of bits used to represent the distance to the last keyframe, and thus how the granulepos field must be shifted or masked to obtain the two parts.

Since libtheora returns compressed data in an `ogg_packet` structure, this may be generally useful even if the Theora packets are not being used in an Ogg container.

Parameters:

- ti* A previously initialized `theora_info`(p. 13) struct

Returns:

The bit shift dividing the two granulepos fields

This function was added in the 1.0alpha5 release.

7.1.4.20 `double theora_granule_time (theora_state * th, ogg_int64_t granulepos)`

Convert a granulepos to absolute time in seconds.

The granulepos is interpreted in the context of a given `theora_state`(p. 15) handle.

Parameters:

- th* A previously initialized `theora_state`(p. 15) handle (encode or decode)
- granulepos* The granulepos to convert.

Returns:

The absolute time in seconds corresponding to *granulepos*.

Return values:

- 1. The given granulepos is undefined (i.e. negative), or
- 1. The function has been disabled because floating point support is not available.

7.1.4.21 void theora_info_clear (theora_info * c)

Clear a **theora_info**(p. 13) structure.

All values within the given **theora_info**(p. 13) structure are cleared, and associated internal codec setup data is freed.

Parameters:

c A **theora_info**(p. 13) struct to initialize.

7.1.4.22 void theora_info_init (theora_info * c)

Initialize a **theora_info**(p. 13) structure.

All values within the given **theora_info**(p. 13) structure are initialized, and space is allocated within libtheora for internal codec setup data.

Parameters:

c A **theora_info**(p. 13) struct to initialize.

7.1.4.23 int theora_packet_isheader (ogg_packet * op)

Report whether a theora packet is a header or not This function does no verification beyond checking the header flag bit so it should not be used for bitstream identification; use **theora_decode_header**(p. 22) for that.

Parameters:

op An ogg_packet containing encoded theora data.

Return values:

- 1 The packet is a header packet
- 0 The packet is not a header packet (and so contains frame data)

Thus function was added in the 1.0alpha4 release.

7.1.4.24 int theora_packet_iskeyframe (ogg_packet * op)

Report whether a theora packet is a keyframe or not.

Parameters:

op An ogg_packet containing encoded theora data.

Return values:

- 1* The packet contains a keyframe image
- 0* The packet is contains an interframe delta
- 1* The packet is not an image data packet at all

This function was added in the 1.0alpha4 release.

7.1.4.25 `ogg_uint32_t theora_version_number (void)`

Retrieve a 32-bit version number.

This number is composed of a 16-bit major version, 8-bit minor version and 8 bit sub-version, composed as follows:

$$(\text{VERSION_MAJOR} \ll 16) + (\text{VERSION_MINOR} \ll 8) + (\text{VERSION_SUB})$$
Returns:

The version number.

7.1.4.26 `const char* theora_version_string (void)`

Retrieve a human-readable string to identify the encoder vendor and version.

Returns:

A version string.

Index

- `/home/giles/projects/xiph/theora/include/`
 - Directory Reference, 9
 - `/home/giles/projects/xiph/theora/include/theora/`
 - Directory Reference, 10
- OC_CS_ITU_REC_470BG
 - theora.h, 20
- OC_CS_ITU_REC_470M
 - theora.h, 20
- OC_CS_NSAPACES
 - theora.h, 20
- OC_CS_UNSPECIFIED
 - theora.h, 20
- OC_PF_420
 - theora.h, 20
- OC_PF_422
 - theora.h, 20
- OC_PF_444
 - theora.h, 20
- OC_PF_RSVD
 - theora.h, 20
- theora.h, 17
 - OC_CS_ITU_REC_470BG, 20
 - OC_CS_ITU_REC_470M, 20
 - OC_CS_NSAPACES, 20
 - OC_CS_UNSPECIFIED, 20
 - OC_PF_420, 20
 - OC_PF_422, 20
 - OC_PF_444, 20
 - OC_PF_RSVD, 20
 - theora_clear, 20
 - theora_colorspace, 20
 - theora_comment, 20
 - theora_comment_add, 21
 - theora_comment_add_tag, 21
 - theora_comment_clear, 21
 - theora_comment_init, 21
 - theora_comment_query, 21
 - theora_comment_query_count, 22
 - theora_decode_header, 22
 - theora_decode_init, 23
 - theora_decode_packetin, 23
 - theora_decode_YUVout, 23
 - theora_encode_comment, 24
- theora_encode_header, 24
 - theora_encode_init, 24
 - theora_encode_packetout, 24
 - theora_encode_tables, 25
 - theora_encode_YUVin, 25
 - theora_granule_frame, 25
 - theora_granule_shift, 26
 - theora_granule_time, 26
 - theora_info_clear, 27
 - theora_info_init, 27
 - theora_packet_isheader, 27
 - theora_packet_iskeyframe, 27
 - theora_pixelformat, 20
 - theora_version_number, 28
 - theora_version_string, 28
- theora_clear
 - theora.h, 20
- theora_colorspace
 - theora.h, 20
- theora_comment, 11
 - theora.h, 20
- theora_comment_add
 - theora.h, 21
- theora_comment_add_tag
 - theora.h, 21
- theora_comment_clear
 - theora.h, 21
- theora_comment_init
 - theora.h, 21
- theora_comment_query
 - theora.h, 21
- theora_comment_query_count
 - theora.h, 22
- theora_decode_header
 - theora.h, 22
- theora_decode_init
 - theora.h, 23
- theora_decode_packetin
 - theora.h, 23
- theora_decode_YUVout
 - theora.h, 23
- theora_encode_comment
 - theora.h, 24
- theora_encode_header
 - theora.h, 24

theora_encode_init
 theora.h, 24

theora_encode_packetout
 theora.h, 24

theora_encode_tables
 theora.h, 25

theora_encode_YUVin
 theora.h, 25

theora_granule_frame
 theora.h, 25

theora_granule_shift
 theora.h, 26

theora_granule_time
 theora.h, 26

theora_info, 13

theora_info_clear
 theora.h, 27

theora_info_init
 theora.h, 27

theora_packet_isheader
 theora.h, 27

theora_packet_iskeyframe
 theora.h, 27

theora_pixelformat
 theora.h, 20

theora_state, 15

theora_version_number
 theora.h, 28

theora_version_string
 theora.h, 28

yuv_buffer, 16