libmad

0.15.2b

Generated by Doxygen 1.5.5

Tue Jun 10 13:57:06 2008

Contents

Chapter 1

libmad - MPEG audio decoder library

Author:

Except where otherwise noted, all code was authored by: Robert Leslie <rob at underbit com>
Copyright (C) 2000-2008 Underbit Technologies, Inc.

See the file CREDITS for a list of contributors.

1.1 Introduction

MAD (libmad) is a high-quality MPEG audio decoder. It currently supports MPEG-1 and the MPEG-2 extension to Lower Sampling Frequencies, as well as the so-called MPEG 2.5 format. All three audio layers (Layer I, Layer II, and Layer III a.k.a. MP3) are fully implemented.

MAD does not yet support MPEG-2 multichannel audio (although it should be backward compatible with such streams) nor does it currently support AAC.

MAD has the following special features:

- 24-bit PCM output
- 100% fixed-point (integer) computation
- completely new implementation based on the ISO/IEC standards
- distributed under the terms of the GNU General Public License (GPL)

Because MAD provides full 24-bit PCM output, applications using MAD are able to produce high quality audio. Even when the output device supports only 16-bit PCM, applications can use the extra resolution to increase the audible dynamic range through the use of dithering or noise shaping.

Because MAD uses integer computation rather than floating point, it is well suited for architectures without a floating point unit. All calculations are performed with a 32-bit fixed-point integer representation.

Because MAD is a new implementation of the ISO/IEC standards, it is unencumbered by the errors of other implementations. MAD is NOT a derivation of the ISO reference source or any other code. Considerable effort has been expended to ensure a correct implementation, even in cases where the standards are ambiguous or misleading.

Because MAD is distributed under the terms of the GPL, its redistribution is not generally restricted, so long as the terms of the GPL are followed. This means MAD can be incorporated into other software as long as that software is also distributed under the GPL. (Should this be undesirable, alternate arrangements may be possible by contacting Underbit.)

1.2 About the code

The code is optimized and performs very well, although specific improvements can still be made. The output from the decoder library consists of 32-bit signed linear fixed-point values that can be easily scaled for any size PCM output, up to 24 bits per sample.

The API for libmad can be found in the mad.h header file. Note that this file is automatically generated, from the following files:

- version.h
- fixed.h
- bit.h
- timer.h
- stream.h
- frame.h
- synth.h
- decoder.h

so it will not exist until after you have built the library.

There are two APIs available, one High-Level API, and the other Low-Level API. With the low-level API, each step of the decoding process must be handled explicitly, offering the greatest amount of control. With the high-level API, after callbacks are configured, a single routine will decode an entire bitstream.

1.2.1 Demo

The file minimad.c contains an example usage of the libmad API that shows only the bare minimum required to implement a useful decoder. It expects a regular file to

be redirected to standard input, and it sends decoded 16-bit signed little-endian PCM samples standard error and decoding continues. Note that the scale() to standard output. If a decoding error occurs, it is reported to routine in this code is only provided as an example; it rounds MAD's high-resolution samples down to 16 bits, but does not perform any dithering or noise shaping. It is therefore not recommended to use this routine as-is in your own code if sound quality is important.

1.3 Integer Performance

To get the best possible performance, it is recommended that an assembly version of the fixed-point multiply and related routines be selected. Several such assembly routines have been written for various CPUs.

If an assembly version is not available, a fast approximation version will be used. This will result in reduced accuracy of the decoder.

Alternatively, if 64-bit integers are supported as a datatype by the compiler, another version can be used that is much more accurate. However, using an assembly version is generally much faster and just as accurate.

More information can be gathered from the ' fixed.h ' header file.

MAD's CPU-intensive subband synthesis routine can be further optimized at the expense of a slight loss in output accuracy due to a modified method for fixed-point multiplication with a small windowing constant. While this is helpful for performance and the output accuracy loss is generally undetectable, it is disabled by default and must be explicitly enabled.

Under some architectures, other special optimizations may also be available.

1.3.1 Audio Quality

The output from MAD has been found to satisfy the ISO/IEC 11172-4 computational accuracy requirements for compliance. In most configurations, MAD is a Full Layer III ISO/IEC 11172-3 audio decoder as defined by the standard.

When the approximation version of the fixed-point multiply is used, MAD is a limited accuracy ISO/IEC 11172-3 audio decoder as defined by the standard.

MAD can alternatively be configured to produce output with less or more accuracy than the default, as a tradeoff with performance.

MAD produces output samples with a precision greater than 24 bits. Because most output formats use fewer bits, typically 16, it is recommended that a dithering algorithm be used (rather than rounding or truncating) to obtain the highest quality audio. However, dithering may unfavorably affect an analytic examination of the output (such as compliance testing); you may therefore wish to use rounding in this case instead.

1.3.2 Portability Issues

GCC is preferred to compile the code, but other compilers may also work. The assembly code in ' fixed.h ' depends on the inline assembly features of your compiler. If you're not using GCC or MSVC++, you can either write your own assembly macros or use the default (low quality output) version.

The union initialization of ' huffman.c ' may not be portable to all platforms when GCC is not used.

The code should not be sensitive to word sizes or byte ordering, however it does assume A % B has the same sign as A.

1.4 Building and Installing

1.4.1 Windows Platforms

MAD can be built under Windows using either MSVC++ or Cygwin. A MSVC++ project file can be found under the ' msvc++ ' subdirectory.

To build libmad using Cygwin, you will first need to install the Cygwin tools:

http://www.cygwin.com/

You may then proceed with the following POSIX instructions within the Cygwin shell.

Note that by default Cygwin will build a library that depends on the Cygwin DLL. You can use MinGW to build a library that does not depend on the Cygwin DLL. To do so, give the option –host=mingw32 to ' configure '.

1.4.2 POSIX Platforms (including Cygwin)

The code is distributed with a ' configure ' script that will generate for you a ' Makefile ' and a ' config.h ' for your platform. See the file ' INSTALL ' for generic instructions.

The specific options you may want to give ' configure ' are:

–enable-speed optimize for speed over accuracy

–enable-accuracy optimize for accuracy over speed

–disable-debugging do not compile with debugging support, and use more optimizations

–disable-shared do not build a shared library

Note that you need not specify one of –enable-speed or –enable-accuracy; in its default configuration, MAD is optimized for both. You should only use one of these options if you wish to compromise speed or accuracy for the other.

By default the package will build a shared library if possible for your platform. If you want only a static library, use –disable-shared.

It is not normally necessary to use the following options, but you may fine-tune the configuration with them if desired:

–enable-fpm=ARCH use the ARCH-specific version of the fixed-point math assembly routines (current options are: intel, arm, mips, sparc, ppc; also allowed are: 64bit, approx)

–enable-sso use the subband synthesis optimization, with reduced accuracy

–disable-aso do not use certain architecture-specific optimizations

By default an appropriate fixed-point assembly routine will be selected for the configured host type, if it can be determined. Thus if you are cross-compiling for another architecture, you should be sure either to give ‘ configure ’ a host type argument (–host) or to use an explicit –enable-fpm option.

If an appropriate assembly routine cannot be determined, the default approximation version will be used. In this case, use of an alternate –enable-fpm is highly recommended.

1.4.3 Experimenting and Developing

Further options for ‘ configure ’ that may be useful to developers and experimenters are:

–enable-debugging enable diagnostic debugging support and debugging symbols

–enable-profiling generate gprof profiling code

–enable-experimental enable code using the EXPERIMENTAL preprocessor define

1.5 Copyright

Please read the ‘ COPYRIGHT ’ file for copyright and warranty information. Also, the file ‘ COPYING ’ contains the full text of the GNU GPL.

Send inquiries, comments, bug reports, suggestions, patches, etc. to:

Underbit Technologies, Inc. <support at underbit com>

See also the MAD home page on the Web:

http://www.underbit.com/products/mad/

Chapter 2

High-Level API

With the high-level API, after callbacks are configured, a single routine will decode an entire bitstream.

Joe Drew < hoserhead at woot net > posted this interesting message on maddev mailing list: http://www.mars.org/mailman/public/mad-dev/2001-October/000369.html

Most of the following is picked from it.

2.1 basic information

MAD operates with `callbacks` for functions. Each of these functions is expected to return type enum mad_flow ; this allows you to control the decoding process.

MAD always outputs mad_fixed_t data (which is only guaranteed to be ∗at least∗ 32 bits wide, and in the same endianness as the native integer types). Take this into account when outputting samples to the sound card.

Related to the above, since MAD outputs type mad_fixed_t, unless you can output with 32-bit accuracy (most sound cards can't), you will have to quantize, round, dither, etc these samples to 16-bit (or whatever you need.) While there is a sample routine in minimad.c, if you want good quality you'll either want to roll your own or take a look in madplay's sources.

2.2 Integral to understanding MAD

MAD is a decoding library only. You handle input and output; you're responsible for fast-forwarding and rewinding, if you want that type of functionality. All that MAD will do is take input from you, decode the MPEG frames, give you some information about them, and give you the decoded PCM data.

2.3 nitty-gritty information

First, you need a mad_decoder struct. This holds all information about how you want your stream decoded, such as input/output functions, error handling functions, etc.

mad_decoder_init() sets this structure up for you.

```
struct mad_decoder decoder;
struct my_playbuf playbuf;

mad_decoder_init(&decoder, &playbuf, input_func, header_func,
                 /*filter*/ 0, output_func, /*error*/ 0, /* message */ 0);
```

In this example, the function called to get more data is set to input_callback, the function called after MPEG headers have been decoded is header_func, the function called after all sound data has been decoded to PCM (for output) is output_callback, and the filter, error, and message functions are unset.

Now, MAD runs in a constant decoding loop. It runs something along the following lines:

```
if I'm out of data
 call input_func
if input_func says there's no more data,
 quit
decode the header and call header_func
decode the mpeg audio data
call the filter function
call the output function
loop
```

Now, this is an oversimplification obviously. The important thing to realise is that at every step of the process you can tell MAD what to do.

Since all of these functions return enum mad_flow, you can tell MAD to do any of the following:

MAD_FLOW_CONTINUE Keep decoding this stream

MAD_FLOW_STOP Stop decoding this stream, but exit normally

MAD_FLOW_BREAK Stop decoding this stream, and exit with an error

MAD_FLOW_IGNORE Don't decode this frame, but continue afterwards

Most of the time you'll probably want to return MAD_FLOW_CONTINUE. In every case, you'll have to return one of these values from the functions you define.

This is the definition of each of the functions:

```
enum mad_flow (*input_func)(void *, struct mad_stream *);
enum mad_flow (*header_func)(void *, struct mad_header const *);
enum mad_flow (*filter_func)(void *, struct mad_stream const *, struct
mad_frame *);
enum mad_flow (*output_func)(void *, struct mad_header const *, struct
mad_pcm *);
enum mad_flow (*error_func)(void *, struct mad_stream *, struct
mad_frame *);
enum mad_flow (*message_func)(void *, void *, unsigned int *);
```

In each of these functions the *void∗ pointer* passed to the function is your "playbuf" structure. This can hold whatever you want - for example, song title, length, number of frames - just remember to re-cast it to the type you've defined.

input_func takes a mad_stream pointer. Most of the time what you'll want to do is something along the lines of the following:

```
if (more_data_available)
  buffer = refill_buffer();
  mad_stream_buffer(stream, buffer, length_of_buffer);
  return MAD_FLOW_CONTINUE;
else
  return MAD_FLOW_STOP;
```

(On many systems you'll want to use mmap() for this.)

header_func takes a mad_header pointer. This contains most of the important information about a given frame; in constant bitrate files, it can contain most of the important information about the stream. It will give you the length of that frame, using mad_timer_t; the audio layer; extension; bitrate... the list is long. Read frame.h or mad.h in the frame.h area for more information. Again, return MAD_FLOW_-{CONTINUE,STOP,BREAK} depending on outside conditions.

The only other function I have firsthand information on is output_func; in this case, you are given a pointer to `struct mad_pcm`. This gives you the sampling rate, number of channels, and number of samples per channel; doing something like the following should work:

```
mad_fixed_t *left_channel = pcm->samples[0], *right_channel =
pcm->samples[1];
int nsamples = pcm->length;
signed int sample;
unsigned char * buffer = some_buffer;
unsigned char * ptr = buffer;

while (nsamples--)
{
            sample = (signed int) do_downsample(*left_ch++)

            *ptr++ = (unsigned char) (sample >> 0);
            *ptr++ = (unsigned char) (sample >> 8);

            sample = (signed int) do_downsample(*right_ch++)

            *ptr++ = (unsigned char) (sample >> 0);
            *ptr++ = (unsigned char) (sample >> 8);
}

output buffer to device.
```

Be sure to handle the big-endian case (autoconf can test for this), and also the mono (1 channel) case. See `mad.c in mpg321`, at the end of the file, for an example.

Information on the other (error, filter, message) functions would be appreciated, though I think in knowing this information anyone should be able to puzzle it out.

Now that the decoder is set up with all these callback functions, you call

```
mad_decoder_run(&decoder, MAD_DECODER_MODE_SYNC);
```

and then

```
mad_decoder_finish(&decoder);
```

Once you've called mad_decoder_finish, you can re-use the decoder struct, if you're, for example, within a playlist. Incidentally, all MAD structures have similar `mad_-(whatever)_init` and `mad_(whatever)_finish` functions.

I hope this helps people get their feet wet with MAD. Read the source, and particularly mad.h - there are a lot of things there you might not expect. Rob has done a good job in making MAD a complete solution. :)

See also:

Low-Level API

Chapter 3

Low-Level API

With the low-level API, each step of the decoding process must be handled explicitly, offering the greatest amount of control.

By way of clarification, MAD also has a low-level API which does not use callbacks. You can control the entire decoding process yourself more or less as follows:

```
/* load buffer with your MPEG audio data */

mad_stream_buffer(&stream, buffer, buflen);

while (1) {
  mad_frame_decode(&frame, &stream);
  mad_synth_frame(&synth, &frame);

  /* output PCM samples in synth.pcm */
}
```

This is vastly simplified, but it shows the general idea. mad_frame_decode() decodes the next frame's header and subband samples. mad_synth_frame() takes those subband samples and synthesizes PCM samples.

It is also possible to call mad_header_decode() before mad_frame_decode(). This just gives you the frame's header info, in case that's all you want, or perhaps to help you decide whether you want to decode the rest of the frame.

3.1 Initialisation and cleanup

As Joe mentions, each of the stream, frame, and synth structs needs to be initialized and "finished" before and after use:

```
struct mad_stream stream;
struct mad_frame frame;
struct mad_synth synth;

mad_stream_init(&stream);
mad_frame_init(&frame);
mad_synth_init(&synth);

/* ... */

mad_synth_finish(&synth);
mad_frame_finish(&frame);
mad_stream_finish(&stream);
```

You can work with just a struct mad_header instead of a struct mad_frame if you only want to decode frame headers.

3.2 Converting MAD's fixed-point integer samples

The fixed-point sample format is important to understand, and I recommend reading the comments in libmad/fixed.h.

The thing to remember when converting MAD's fixed-point integer samples to 16-bit PCM (or whatever) is that MAD encodes samples as numbers in the full-scale range [-1.0, +1.0) where the binary point is placed 28 (MAD_F_FRACBITS) bits to the left of the integer.

However, you need to be prepared to handle clipping as some numbers may be less than -1.0 (-MAD_F_ONE) or greater than or equal to +1.0 (MAD_F_ONE, aka 1 << MAD_F_FRACBITS).

See also:

High-Level API

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Module Documentation

7.1 Layer I

Functions

- static mad_fixed_t I_sample (struct mad_bitptr *ptr, unsigned int nb)

  Decodes one requantized Layer I sample from a bitstream.

- int mad_layer_I (struct mad_stream *stream, struct mad_frame *frame)

  Decodes a single Layer I frame.

Variables

- static mad_fixed_t const linear_table [14]

  linear scaling table

7.1.1 Function Documentation

7.1.1.1 static mad_fixed_t I_sample (struct mad_bitptr * *ptr*, unsigned int *nb*) [static]

Decodes one requantized Layer I sample from a bitstream.

Definition at line 80 of file layer12.c.

References linear_table, mad_bit_read(), MAD_F_FRACBITS, mad_f_mul, and MAD_F_ONE.

Referenced by mad_layer_I().

Here is the call graph for this function:

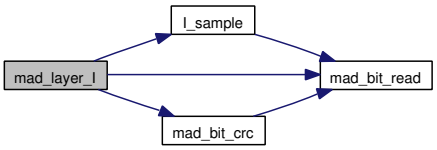
7.1.1.2 int mad_layer_I (struct mad_stream * *stream*, struct mad_frame * *frame*)

Decodes a single Layer I frame.

Definition at line 108 of file layer12.c.

References mad_header::crc_check, mad_header::crc_target, mad_stream::error, mad_header::flags, mad_frame::header, I_sample(), mad_bit_crc(), mad_bit_read(), MAD_ERROR_BADBITALLOC, MAD_ERROR_BADCRC, MAD_ERROR_BADSCALEFACTOR, mad_f_mul, MAD_FLAG_I_STEREO, MAD_FLAG_PROTECTION, MAD_MODE_JOINT_STEREO, MAD_NCHANNELS, MAD_OPTION_IGNORECRC, mad_header::mode, mad_header::mode_extension, mad_frame::options, mad_stream::ptr, s, mad_frame::sbsample, and sf_table.

Here is the call graph for this function:



7.1.2 Variable Documentation

7.1.2.1 mad_fixed_t const linear_table[14] [static]

Initial value:

```
{
 MAD_F(0x15555555),
 MAD_F(0x12492492),
 MAD_F(0x11111111),
 MAD_F(0x10842108),
 MAD_F(0x10410410),
 MAD_F(0x10204081),
 MAD_F(0x10101010),
 MAD_F(0x10080402),
 MAD_F(0x10040100),
 MAD_F(0x10020040),
 MAD_F(0x10010010),
 MAD_F(0x10008004),
 MAD_F(0x10004001),
 MAD_F(0x10002000)
}
```

linear scaling table

Definition at line 59 of file layer12.c.

Referenced by I_sample().

7.2 Layer II

Data Structures

- struct quantclass

  quantization class table

Functions

- static void II_samples (struct mad_bitptr *ptr, struct quantclass const *quantclass, mad_fixed_t output[3])

  Decodes three requantized Layer II samples from a bitstream.

- int mad_layer_II (struct mad_stream *stream, struct mad_frame *frame)

  Decodes a single Layer II frame.

Variables

- struct {
  unsigned int sblimit
  unsigned char const offsets [30]
  } sbquant_table [5]

- struct {
  unsigned short nbal
  unsigned short offset
  } bitalloc_table [8]

- static unsigned char const offset_table [6][15]

  offsets into quantization class table

- static struct quantclass qc_table [17]

7.2.1 Function Documentation

7.2.1.1 static void II_samples (struct mad_bitptr * *ptr*, struct quantclass const * *quantclass*, mad_fixed_t *output*[3]) [static]

Decodes three requantized Layer II samples from a bitstream.

Definition at line 292 of file layer12.c.

References quantclass::bits, quantclass::C, quantclass::D, quantclass::group, mad_bit_read(), MAD_F_FRACBITS, mad_f_mul, quantclass::nlevels, and s.

Referenced by mad_layer_II().

Here is the call graph for this function:

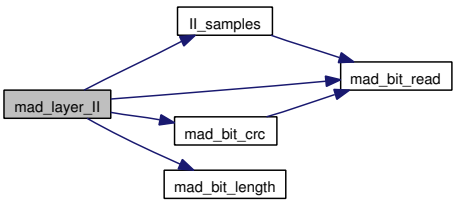
7.2.1.2 int mad_layer_II (struct mad_stream * *stream*, struct mad_frame * *frame*)

Decodes a single Layer II frame.

Definition at line 341 of file layer12.c.

References bitalloc_table, mad_header::bitrate, mad_header::crc_check, mad_header::crc_target, mad_stream::error, mad_header::flags, mad_frame::header, II_samples(), mad_bit_crc(), mad_bit_length(), mad_bit_read(), MAD_ERROR_BADCRC, MAD_ERROR_BADMODE, MAD_ERROR_BADSCALEFACTOR, mad_f_mul, MAD_FLAG_FREEFORMAT, MAD_FLAG_I_STEREO, MAD_FLAG_LSF_EXT, MAD_FLAG_PROTECTION, MAD_MODE_JOINT_STEREO, MAD_NCHANNELS, MAD_OPTION_IGNORECRC, mad_header::mode, mad_header::mode_extension, nbal, offset_table, offsets, mad_frame::options, mad_stream::ptr, qc_table, s, mad_header::samplerate, sblimit, sbquant_table, mad_frame::sbsample, and sf_table.

Here is the call graph for this function:



7.2.2 Variable Documentation

7.2.2.1 struct { ... } bitalloc_table[8] [static]

Referenced by mad_layer_II().

7.2.2.2 unsigned { ... } nbal [inherited]

Definition at line 252 of file layer12.c.

7.2.2.3 unsigned { ... } offset [inherited]

Definition at line 253 of file layer12.c.

7.2.2.4 unsigned char const offset_table[6][15] [static]

Initial value:

```
 {
  { 0, 1, 16                                                    },
  { 0, 1,  2, 3, 4, 5, 16                                       },
  { 0, 1,  2, 3, 4, 5,  6, 7,  8,  9, 10, 11, 12, 13, 14 },
  { 0, 1,  3, 4, 5, 6,  7, 8,  9, 10, 11, 12, 13, 14, 15 },
  { 0, 1,  2, 3, 4, 5,  6, 7,  8,  9, 10, 11, 12, 13, 16 },
  { 0, 2,  4, 5, 6, 7,  8, 9, 10, 11, 12, 13, 14, 15, 16 }
}
```

offsets into quantization class table

Definition at line 267 of file layer12.c.

Referenced by mad_layer_II().

7.2.2.5 unsigned { ... } offsets[30] [inherited]

Definition at line 232 of file layer12.c.

7.2.2.6 struct quantclass qc_table[17] [static]

Referenced by mad_layer_II().

7.2.2.7 unsigned { ... } sblimit [inherited]

Definition at line 231 of file layer12.c.

7.2.2.8 struct { ... } sbquant_table[5] [static]

Referenced by mad_layer_II().

7.3 Layer III

Data Structures

- struct sideinfo
- struct fixedfloat

  table for requantization.

Defines

- #define sfb_16000_long sfb_22050_long
- #define sfb_12000_long sfb_16000_long

  MPEG 2.5 scalefactor band widths.

- #define sfb_11025_long sfb_12000_long
- #define sfb_12000_short sfb_16000_short
- #define sfb_11025_short sfb_12000_short
- #define sfb_12000_mixed sfb_16000_mixed
- #define sfb_11025_mixed sfb_12000_mixed
- #define MASK(cache, sz, bits) (((cache) >> ((sz) - (bits))) & ((1 << (bits)) - 1))
- #define MASK1BIT(cache, sz) ((cache) & (1 << ((sz) - 1)))

Enumerations

- enum { count1table_select = 0x01, scalefac_scale = 0x02, preflag = 0x04, mixed_block_flag = 0x08 }
- enum { I_STEREO = 0x1, MS_STEREO = 0x2 }

Functions

- static enum mad_error III_sideinfo (struct mad_bitptr *ptr, unsigned int nch, int lsf, struct sideinfo *si, unsigned int *data_bitlen, unsigned int *priv_bitlen)

  decode frame side information from a bitstream.

- static unsigned int III_scalefactors_lsf (struct mad_bitptr *ptr, struct channel *channel, struct channel *gr1ch, int mode_extension)

  decode channel scalefactors for LSF from a bitstream.

- static unsigned int III_scalefactors (struct mad_bitptr *ptr, struct channel *channel, struct channel const *gr0ch, unsigned int scfsi)

  decode channel scalefactors of one granule from a bitstream.

- static void III_exponents (struct channel const *channel, unsigned char const *sfbwidth, signed int exponents[39])

The Layer III formula for requantization and scaling is defined by section 2.4.3.4.7.1 of ISO/IEC 11172-3, as follows:.

- static mad_fixed_t III_requantize (unsigned int value, signed int exp)

  requantize one (positive) value.

- static enum mad_error III_huffdecode (struct mad_bitptr *ptr, mad_fixed_t xr[576], struct channel *channel, unsigned char const *sfbwidth, unsigned int part2_length)

  decode Huffman code words of one channel of one granule.

- static void III_reorder (mad_fixed_t xr[576], struct channel const *channel, unsigned char const sfbwidth[39])

  reorder frequency lines of a short block into subband order.

- static enum mad_error III_stereo (mad_fixed_t xr[2][576], struct granule const *granule, struct mad_header *header, unsigned char const *sfbwidth)

  perform joint stereo processing on a granule.

- static void III_aliasreduce (mad_fixed_t xr[576], int lines)

  perform frequency line alias reduction.

- static void fastsdct (mad_fixed_t const x[9], mad_fixed_t y[18])
- static void sdctII (mad_fixed_t const x[18], mad_fixed_t X[18])
- static void dctIV (mad_fixed_t const y[18], mad_fixed_t X[18])
- static void imdct36 (mad_fixed_t const x[18], mad_fixed_t y[36])

  perform X[18]->x[36] IMDCT using Szu-Wei Lee's fast algorithm.

- static void III_imdct_l (mad_fixed_t const X[18], mad_fixed_t z[36], unsigned int block_type)

  perform IMDCT and windowing for long blocks.

- static void III_imdct_s (mad_fixed_t const X[18], mad_fixed_t z[36])

  perform IMDCT and windowing for short blocks.

- static void III_overlap (mad_fixed_t const output[36], mad_fixed_t overlap[18], mad_fixed_t sample[18][32], unsigned int sb)

  perform overlap-add of windowed IMDCT outputs.

- static void III_overlap_z (mad_fixed_t overlap[18], mad_fixed_t sample[18][32], unsigned int sb)

  perform "overlap-add" of zero IMDCT outputs.

- static void III_freqinver (mad_fixed_t sample[18][32], unsigned int sb)

  perform subband frequency inversion for odd sample lines.

- static enum mad_error III_decode (struct mad_bitptr *ptr, struct mad_frame *frame, struct sideinfo *si, unsigned int nch)

  decode frame main_data.

- int mad_layer_III (struct mad_stream *stream, struct mad_frame *frame)

  decode a single Layer III frame.

Variables

- struct {
  unsigned char slen1
  unsigned char slen2
  } sflen_table [16]

- static unsigned char const nsfb_table [6][3][4]

  number of LSF scalefactor band values.

- static unsigned char const sfb_48000_long [ ]

  MPEG-1 scalefactor band widths.

- static unsigned char const sfb_44100_long [ ]
- static unsigned char const sfb_32000_long [ ]
- static unsigned char const sfb_48000_short [ ]
- static unsigned char const sfb_44100_short [ ]
- static unsigned char const sfb_32000_short [ ]
- static unsigned char const sfb_48000_mixed [ ]
- static unsigned char const sfb_44100_mixed [ ]
- static unsigned char const sfb_32000_mixed [ ]
- static unsigned char const sfb_24000_long [ ]

  MPEG-2 scalefactor band widths.

- static unsigned char const sfb_22050_long [ ]
- static unsigned char const sfb_24000_short [ ]
- static unsigned char const sfb_22050_short [ ]
- static unsigned char const sfb_16000_short [ ]
- static unsigned char const sfb_24000_mixed [ ]
- static unsigned char const sfb_22050_mixed [ ]
- static unsigned char const sfb_16000_mixed [ ]
- static unsigned char const sfb_8000_long [ ]
- static unsigned char const sfb_8000_short [ ]
- static unsigned char const sfb_8000_mixed [ ]
- struct {
  unsigned char const * l
  unsigned char const * s
  unsigned char const * m
  } sfbwidth_table [9]

- static unsigned char const pretab [22]

  scalefactor band preemphasis (used only when preflag is set).

- static struct fixedfloat rq_table [8207]
- static mad_fixed_t const root_table [7]

  fractional powers of two.

- static mad_fixed_t const cs [8]

  coefficients for aliasing reduction.

- static mad_fixed_t const ca [8]
- static mad_fixed_t const imdct_s [6][6]

  IMDCT coefficients for short blocks.

- static mad_fixed_t const window_l [36]

  windowing coefficients for long blocks.

- static mad_fixed_t const window_s [12]

  windowing coefficients for short blocks.

- static mad_fixed_t const is_table [7]

  coefficients for intensity stereo processing.

- static mad_fixed_t const is_lsf_table [2][15]

  coefficients for LSF intensity stereo processing.

7.3.1 Define Documentation

7.3.1.1 #define MASK(cache, sz, bits) (((cache) >> ((sz) - (bits))) & ((1 << (bits)) - 1))

Definition at line 926 of file layer3.c.

Referenced by III_huffdecode().

7.3.1.2 #define MASK1BIT(cache, sz) ((cache) & (1 << ((sz) - 1)))

Definition at line 928 of file layer3.c.

Referenced by III_huffdecode().

7.3.1.3 #define sfb_11025_long sfb_12000_long

Definition at line 275 of file layer3.c.

7.3.1.4 #define sfb_11025_mixed sfb_12000_mixed

Definition at line 294 of file layer3.c.

7.3.1.5 #define sfb_11025_short sfb_12000_short

Definition at line 284 of file layer3.c.

7.3.1.6 #define sfb_12000_long sfb_16000_long

MPEG 2.5 scalefactor band widths.
derived from public sources
Definition at line 274 of file layer3.c.

7.3.1.7 #define sfb_12000_mixed sfb_16000_mixed

Definition at line 293 of file layer3.c.

7.3.1.8 #define sfb_12000_short sfb_16000_short

Definition at line 283 of file layer3.c.

7.3.1.9 #define sfb_16000_long sfb_22050_long

Definition at line 223 of file layer3.c.

7.3.2 Enumeration Type Documentation

7.3.2.1 anonymous enum

Enumerator:

count1table_select
scalefac_scale
preflag
mixed_block_flag

Definition at line 55 of file layer3.c.

7.3.2.2 anonymous enum

Enumerator:

I_STEREO

MS_STEREO

Definition at line 62 of file layer3.c.

7.3.3 Function Documentation

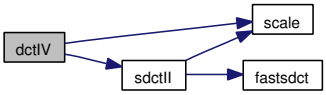
7.3.3.1 static void dctIV (mad_fixed_t const *y*[18], mad_fixed_t *X*[18]) [inline, static]

Definition at line 1692 of file layer3.c.

References MAD_F, mad_f_mul, scale(), and sdctII().

Referenced by imdct36().

Here is the call graph for this function:



7.3.3.2 static void fastsdct (mad_fixed_t const *x*[9], mad_fixed_t *y*[18]) [static]

Definition at line 1579 of file layer3.c.

References MAD_F, and mad_f_mul.

Referenced by sdctII().

7.3.3.3 static void III_aliasreduce (mad_fixed_t *xr*[576], int *lines*) [static]

perform frequency line alias reduction.

Definition at line 1540 of file layer3.c.

References ca, cs, MAD_F_ML0, MAD_F_MLA, and MAD_F_MLZ.

Referenced by III_decode().

7.3.3.4 static enum mad_error III_decode (struct mad_bitptr * *ptr*, struct mad_frame * *frame*, struct sideinfo * *si*, unsigned int *nch*) [static]

decode frame main_data.

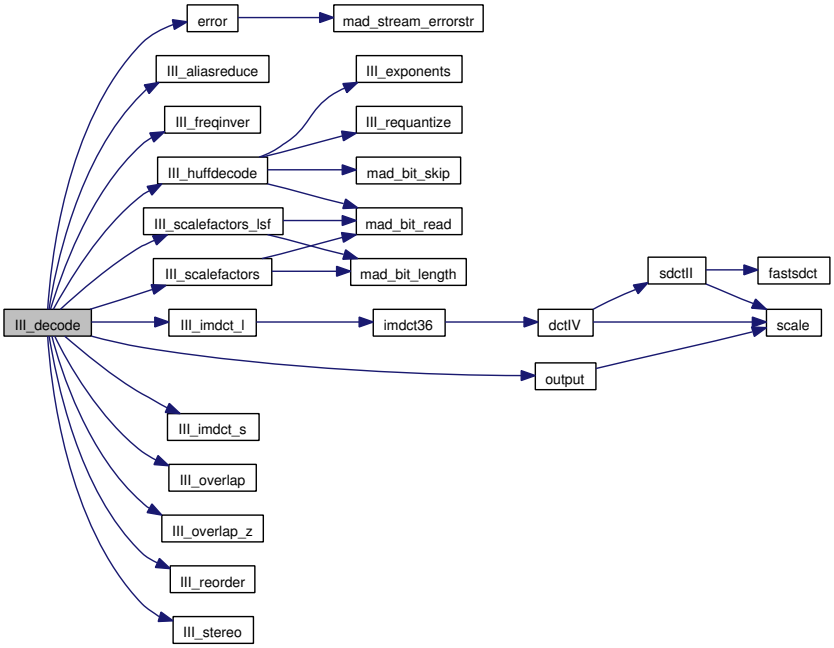
Definition at line 2341 of file layer3.c.

References sideinfo::granule::ch, error(), mad_header::flags, sideinfo::gr, mad_frame::header, III_aliasreduce(), III_freqinver(), III_huffdecode(), III_imdct_l(),

III_imdct_s(), III_overlap(), III_overlap_z(), III_reorder(), III_scalefactors(), III_scalefactors_lsf(), III_stereo(), l, m, MAD_ERROR_NONE, MAD_FLAG_LSF_EXT, MAD_FLAG_MPEG_2_5_EXT, MAD_MODE_JOINT_STEREO, mixed_block_flag, mad_header::mode, mad_header::mode_extension, output(), mad_frame::overlap, s, mad_header::samplerate, sblimit, mad_frame::sbsample, sideinfo::scfsi, and sfbwidth_table.

Referenced by mad_layer_III().

Here is the call graph for this function:



7.3.3.5 static void III_exponents (struct channel const * *channel*, unsigned char const * *sfbwidth*, signed int *exponents*[39]) [static]

The Layer III formula for requantization and scaling is defined by section 2.4.3.4.7.1 of ISO/IEC 11172-3, as follows:.

long blocks: xr[i] = sign(is[i]) * abs(is[i])^(4/3) *
2^((1/4) * (global_gain - 210)) *
2^-(scalefac_multiplier *
(scalefac_l[sfb] + preflag * pretab[sfb]))

short blocks:
```
xr[i] = sign(is[i]) * abs(is[i])^(4/3) *
 2^((1/4) * (global_gain - 210 - 8 * subblock_gain[w])) *
 2^-(scalefac_multiplier * scalefac_s[sfb][w])
```

where:

scalefac_multiplier = (scalefac_scale + 1) / 2

The routines III_exponents() and III_requantize() facilitate this calculation. calculate scalefactor exponents.

Definition at line 816 of file layer3.c.

References l, mixed_block_flag, preflag, pretab, and scalefac_scale.

Referenced by III_huffdecode().

7.3.3.6 static void III_freqinver (mad_fixed_t *sample*[18][32], unsigned int *sb*) [static]

perform subband frequency inversion for odd sample lines.

Definition at line 2308 of file layer3.c.

Referenced by III_decode().

7.3.3.7 static enum mad_error III_huffdecode (struct mad_bitptr * *ptr*, mad_fixed_t *xr*[576], struct channel * *channel*, unsigned char const * *sfbwidth*, unsigned int *part2_length*) [static]

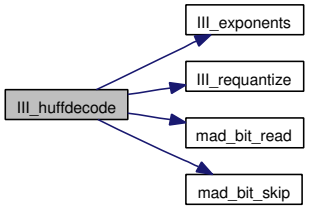
decode Huffman code words of one channel of one granule.

Definition at line 935 of file layer3.c.

References assert, huffquad::bits, huffpair::bits, CHAR_BIT, count1table_select, huffquad::final, huffpair::final, huffquad::hlen, huffpair::hlen, III_exponents(), III_requantize(), hufftable::linbits, mad_bit_bitsleft, mad_bit_read(), mad_bit_skip(), MAD_BUFFER_GUARD, MAD_ERROR_BADHUFFDATA, MAD_ERROR_BADHUFFTABLE, MAD_ERROR_BADPART3LEN, MAD_ERROR_NONE, mad_huff_pair_table, mad_huff_quad_table, MASK, MASK1BIT, huffquad::offset, huffpair::offset, huffquad::ptr, huffpair::ptr, hufftable::startbits, hufftable::table, huffquad::v, huffquad::value, huffpair::value, huffquad::w, huffquad::x, huffpair::x, huffquad::y, and huffpair::y.

Referenced by III_decode().

Here is the call graph for this function:



7.3.3.8 static void III_imdct_l (mad_fixed_t const *X*[18], mad_fixed_t *z*[36], unsigned int *block_type*) [static]

perform IMDCT and windowing for long blocks.

Definition at line 2058 of file layer3.c.

References imdct36(), mad_f_mul, window_l, and window_s.

Referenced by III_decode().

Here is the call graph for this function:

7.3.3.9 static void III_imdct_s (mad_fixed_t const *X*[18], mad_fixed_t *z*[36]) [static]

perform IMDCT and windowing for short blocks.

Definition at line 2144 of file layer3.c.

References imdct_s, MAD_F_ML0, MAD_F_MLA, MAD_F_MLZ, mad_f_mul, s, and window_s.

Referenced by III_decode().

7.3.3.10 static void III_overlap (mad_fixed_t const *output*[36], mad_fixed_t *overlap*[18], mad_fixed_t *sample*[18][32], unsigned int *sb*) [static]

perform overlap-add of windowed IMDCT outputs.

Definition at line 2222 of file layer3.c.

Referenced by III_decode().

7.3.3.11 static void III_overlap_z (mad_fixed_t *overlap*[18], mad_fixed_t *sample*[18][32], unsigned int *sb*) [inline, static]

perform "overlap-add" of zero IMDCT outputs.

Definition at line 2269 of file layer3.c.

Referenced by III_decode().

7.3.3.12 static void III_reorder (mad_fixed_t *xr*[576], struct channel const * *channel*, unsigned char const *sfbwidth*[39]) [static]

reorder frequency lines of a short block into subband order.

Definition at line 1282 of file layer3.c.

References l, mixed_block_flag, and huffquad::w.

Referenced by III_decode().

7.3.3.13 static mad_fixed_t III_requantize (unsigned int *value*, signed int *exp*) [static]

requantize one (positive) value.

Definition at line 886 of file layer3.c.

References CHAR_BIT, fixedfloat::exponent, MAD_F_MAX, mad_f_mul, mad_f_todouble, fixedfloat::mantissa, root_table, and rq_table.

Referenced by III_huffdecode().

7.3.3.14 static unsigned int III_scalefactors (struct mad_bitptr * *ptr*, struct channel * *channel*, struct channel const * *gr0ch*, unsigned int *scfsi*) [static]

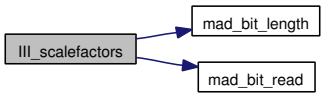
decode channel scalefactors of one granule from a bitstream.

Definition at line 716 of file layer3.c.

References mad_bit_length(), mad_bit_read(), mixed_block_flag, sflen_table, slen1, and slen2.

Referenced by III_decode().

Here is the call graph for this function:



7.3.3.15 static unsigned int III_scalefactors_lsf (struct mad_bitptr * *ptr*, struct channel * *channel*, struct channel * *gr1ch*, int *mode_extension*) [static]

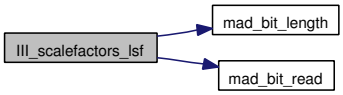
decode channel scalefactors for LSF from a bitstream.

Definition at line 602 of file layer3.c.

References I_STEREO, mad_bit_length(), mad_bit_read(), mixed_block_flag, nsfb_table, and preflag.

Referenced by III_decode().

Here is the call graph for this function:



7.3.3.16 static enum mad_error III_sideinfo (struct mad_bitptr * *ptr*, unsigned int *nch*, int *lsf*, struct sideinfo * *si*, unsigned int * *data_bitlen*, unsigned int * *priv_bitlen*) [static]

decode frame side information from a bitstream.

Definition at line 512 of file layer3.c.

References sideinfo::granule::ch, sideinfo::gr, mad_bit_read(), MAD_ERROR_BADBIGVALUES, MAD_ERROR_BADBLOCKTYPE, MAD_ERROR_BADSCFSI, MAD_ERROR_NONE, sideinfo::main_data_begin, mixed_block_flag, sideinfo::granule::channel::part2_3_length, sideinfo::private_bits, and sideinfo::scfsi.

Referenced by mad_layer_III().

Here is the call graph for this function:

7.3.3.17 static enum mad_error III_stereo (mad_fixed_t *xr*[2][576], struct granule const * *granule*, struct mad_header * *header*, unsigned char const * *sfbwidth*) [static]

perform joint stereo processing on a granule.

Definition at line 1328 of file layer3.c.

References mad_header::flags, I_STEREO, is_lsf_table, is_table, l, m, MAD_ERROR_BADSTEREO, MAD_ERROR_NONE, mad_f_mul, MAD_FLAG_I_STEREO, MAD_FLAG_LSF_EXT, MAD_FLAG_MS_STEREO, mixed_block_flag, mad_header::mode_extension, MS_STEREO, root_table, and s.

Referenced by III_decode().

7.3.3.18 static void imdct36 (mad_fixed_t const *x*[18], mad_fixed_t *y*[36]) [inline, static]

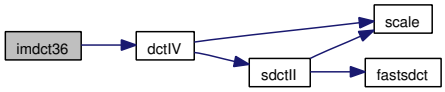
perform X[18]->x[36] IMDCT using Szu-Wei Lee's fast algorithm.

Definition at line 1735 of file layer3.c.

References dctIV().

Referenced by III_imdct_l().

Here is the call graph for this function:



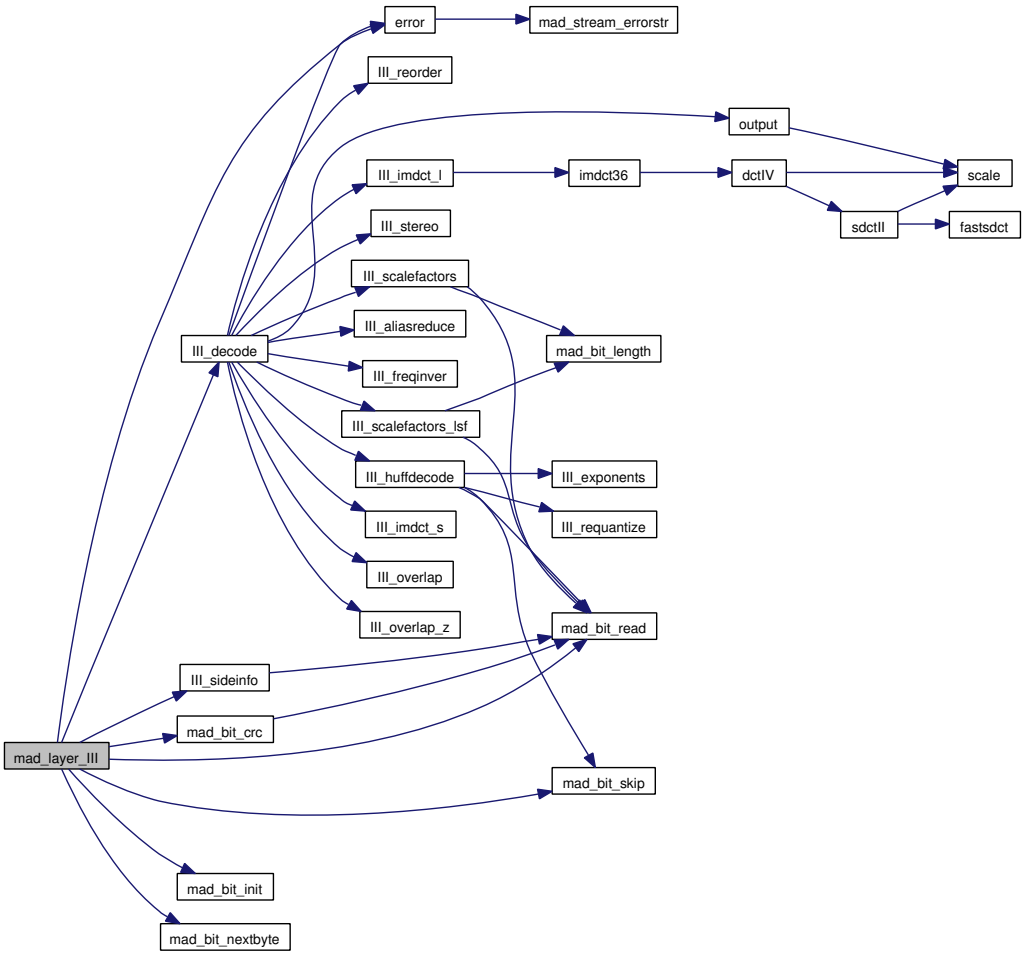
7.3.3.19 int mad_layer_III (struct mad_stream * *stream*, struct mad_frame * *frame*)

decode a single Layer III frame.

Definition at line 2506 of file layer3.c.

References mad_stream::anc_bitlen, mad_stream::anc_ptr, assert, CHAR_BIT, mad_header::crc_check, mad_header::crc_target, mad_stream::error, error(), mad_header::flags, mad_frame::header, III_decode(), III_sideinfo(), mad_bit_crc(), mad_bit_finish, mad_bit_init(), mad_bit_nextbyte(), mad_bit_read(), mad_bit_skip(), MAD_BUFFER_MDLEN, MAD_ERROR_BADCRC, MAD_ERROR_BADDATAPTR, MAD_ERROR_BADFRAMELEN, MAD_ERROR_NOMEM, MAD_FLAG_LSF_EXT, MAD_FLAG_PROTECTION, MAD_NCHANNELS, MAD_OPTION_IGNORECRC, mad_stream::main_data, sideinfo::main_data_begin, mad_stream::md_len, mad_stream::next_frame, mad_frame::options, mad_frame::overlap, sideinfo::private_bits, mad_header::private_bits, and mad_stream::ptr.

Here is the call graph for this function:



7.3.3.20 static void sdctII (mad_fixed_t const *x*[18], mad_fixed_t *X*[18])

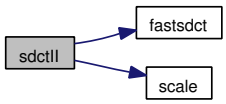
`[inline, static]`

Definition at line 1647 of file layer3.c.

References fastsdct(), MAD_F, mad_f_mul, and scale().

Referenced by dctIV().

Here is the call graph for this function:



7.3.4 Variable Documentation

7.3.4.1 mad_fixed_t const ca[8] [static]

Initial value:

```
 {
  -MAD_F(0x083b5fe7) , -MAD_F(0x078c36d2) ,
  -MAD_F(0x05039814) , -MAD_F(0x02e91dd1) ,
  -MAD_F(0x0183603a) , -MAD_F(0x00a7cb87) ,
  -MAD_F(0x003a2847) , -MAD_F(0x000f27b4)
}
```

Definition at line 379 of file layer3.c.

Referenced by III_aliasreduce().

7.3.4.2 mad_fixed_t const cs[8] [static]

Initial value:

```
 {
  +MAD_F(0x0db84a81) , +MAD_F(0x0e1b9d7f) ,
  +MAD_F(0x0f31adcf) , +MAD_F(0x0fbba815) ,
  +MAD_F(0x0feda417) , +MAD_F(0x0ffc8fc8) ,
  +MAD_F(0x0fff964c) , +MAD_F(0x0ffff8d3)
}
```

coefficients for aliasing reduction.

derived from Table B.9 of ISO/IEC 11172-3

c[] = { -0.6, -0.535, -0.33, -0.185, -0.095, -0.041, -0.0142, -0.0037 } cs[i] = 1 / sqrt(1 + c[i]^2) ca[i] = c[i] / sqrt(1 + c[i]^2)

Definition at line 371 of file layer3.c.

Referenced by III_aliasreduce().

7.3.4.3 mad_fixed_t const imdct_s[6][6] [static]

IMDCT coefficients for short blocks.

derived from section 2.4.3.4.10.2 of ISO/IEC 11172-3

imdct_s[i/even][k] = cos((PI / 24) ∗ (2 ∗ (i / 2) + 7) ∗ (2 ∗ k + 1)) imdct_s[i /odd][k] = cos((PI / 24) ∗ (2 ∗ (6 + (i-1)/2) + 7) ∗ (2 ∗ k + 1))

Definition at line 394 of file layer3.c.

Referenced by III_imdct_s().

7.3.4.4 mad_fixed_t const is_lsf_table[2][15] [static]

coefficients for LSF intensity stereo processing.

derived from section 2.4.3.2 of ISO/IEC 13818-3

is_lsf_table[0][i] = (1 / sqrt(sqrt(2)))^(i + 1) is_lsf_table[1][i] = (1 / sqrt(2)) ^(i + 1)

Definition at line 472 of file layer3.c.

Referenced by III_stereo().

7.3.4.5 mad_fixed_t const is_table[7] [static]

Initial value:

```
 {
  MAD_F(0x00000000) ,
  MAD_F(0x0361962f) ,
  MAD_F(0x05db3d74) ,
  MAD_F(0x08000000) ,
  MAD_F(0x0a24c28c) ,
  MAD_F(0x0c9e69d1) ,
  MAD_F(0x10000000)
}
```

coefficients for intensity stereo processing.

derived from section 2.4.3.4.9.3 of ISO/IEC 11172-3

is_ratio[i] = tan(i ∗ (PI / 12)) is_table[i] = is_ratio[i] / (1 + is_ratio[i])

Definition at line 454 of file layer3.c.

Referenced by III_stereo().

7.3.4.6 unsigned { ... } l [inherited]

Definition at line 308 of file layer3.c.

7.3.4.7 unsigned { ... } m [inherited]

Definition at line 310 of file layer3.c.

7.3.4.8 unsigned char const nsfb_table[6][3][4] [static]

Initial value:

```
 {
  { {  6,  5,  5, 5 },
    {  9,  9,  9, 9 },
    {  6,  9,  9, 9 } },

  { {  6,  5,  7, 3 },
    {  9,  9, 12, 6 },
    {  6,  9, 12, 6 } },

  { { 11, 10,  0, 0 },
```

```
    { 18, 18,  0, 0 },
    { 15, 18,  0, 0 } },

  { {  7,  7,  7, 0 },
    { 12, 12, 12, 0 },
    {  6, 15, 12, 0 } },

  { {  6,  6,  6, 3 },
    { 12,  9,  9, 6 },
    {  6, 12,  9, 6 } },

  { {  8,  8,  5, 0 },
    { 15, 12,  9, 0 },
    {  6, 18,  9, 0 } }
}
```

number of LSF scalefactor band values.

derived from section 2.4.3.2 of ISO/IEC 13818-3

Definition at line 114 of file layer3.c.

Referenced by III_scalefactors_lsf().

7.3.4.9 unsigned char const pretab[22] [static]

Initial value:

```
 {
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 3, 3, 2, 0
}
```

scalefactor band preemphasis (used only when preflag is set).

derived from Table B.6 of ISO/IEC 11172-3

Definition at line 328 of file layer3.c.

Referenced by III_exponents().

7.3.4.10 mad_fixed_t const root_table[7] [static]

Initial value:

```
 {
  MAD_F(0x09837f05) ,
  MAD_F(0x0b504f33) ,
  MAD_F(0x0d744fcd) ,
  MAD_F(0x10000000) ,
  MAD_F(0x1306fe0a) ,
  MAD_F(0x16a09e66) ,
  MAD_F(0x1ae89f99)
}
```

fractional powers of two.

used for requantization and joint stereo decoding

root_table[3 + x] = 2^(x/4)

Definition at line 352 of file layer3.c.

Referenced by III_requantize(), and III_stereo().

7.3.4.11 struct fixedfloat rq_table[8207] [static]

Referenced by III_requantize().

7.3.4.12 unsigned { ... } s [inherited]

Definition at line 309 of file layer3.c.

7.3.4.13 unsigned char const sfb_16000_mixed[ ] [static]

Initial value:

```
 {
     6,  6,  6,  6,  6,  6,
    6,  6,  6,  8,  8,  8, 10, 10, 10, 12,
               12, 12, 14, 14, 14, 18, 18, 18, 24, 24,
               24, 30, 30, 30, 40, 40, 40, 18, 18, 18
}
```

Definition at line 263 of file layer3.c.

7.3.4.14 unsigned char const sfb_16000_short[ ] [static]

Initial value:

```
 {
   4,  4,  4,  4,  4,  4,  4,  4,  4,  6,  6,  6,  8,
   8,  8, 10, 10, 10, 12, 12, 12, 14, 14, 14, 18, 18,
  18, 24, 24, 24, 30, 30, 30, 40, 40, 40, 18, 18, 18
}
```

Definition at line 240 of file layer3.c.

7.3.4.15 unsigned char const sfb_22050_long[ ] [static]

Initial value:

```
 {
   6,  6,  6,  6,  6,  6,  8, 10, 12,  14,  16,
  20, 24, 28, 32, 38, 46, 52, 60, 68,  58,  54
}
```

Definition at line 218 of file layer3.c.

7.3.4.16 unsigned char const sfb_22050_mixed[ ] [static]

Initial value:

```
{
     6,  6,  6,  6,  6,  6,
    6,  6,  6,  6,  6,  6,  8,  8,  8, 10,
              10, 10, 14, 14, 14, 18, 18, 18, 26, 26,
              26, 32, 32, 32, 42, 42, 42, 18, 18, 18
}
```

Definition at line 255 of file layer3.c.

7.3.4.17 unsigned char const sfb_22050_short[ ] [static]

Initial value:

```
{
   4,  4,  4,  4,  4,  4,  4,  4,  4,  6,  6,  6,  6,
   6,  6,  8,  8,  8, 10, 10, 10, 14, 14, 14, 18, 18,
  18, 26, 26, 26, 32, 32, 32, 42, 42, 42, 18, 18, 18
}
```

Definition at line 233 of file layer3.c.

7.3.4.18 unsigned char const sfb_24000_long[ ] [static]

Initial value:

```
{
   6,  6,  6,  6,  6,  6,  8, 10, 12,  14,  16,
  18, 22, 26, 32, 38, 46, 54, 62, 70,  76,  36
}
```

MPEG-2 scalefactor band widths.

derived from Table B.2 of ISO/IEC 13818-3

Definition at line 212 of file layer3.c.

7.3.4.19 unsigned char const sfb_24000_mixed[ ] [static]

Initial value:

```
{
     6,  6,  6,  6,  6,  6,
    6,  6,  6,  8,  8,  8, 10, 10, 10, 12,
              12, 12, 14, 14, 14, 18, 18, 18, 24, 24,
              24, 32, 32, 32, 44, 44, 44, 12, 12, 12
}
```

Definition at line 247 of file layer3.c.

7.3.4.20 unsigned char const sfb_24000_short[ ] [static]

Initial value:

```
 {
   4,  4,  4,  4,  4,  4,  4,  4,  4,  6,  6,  6,  8,
   8,  8, 10, 10, 10, 12, 12, 12, 14, 14, 14, 18, 18,
  18, 24, 24, 24, 32, 32, 32, 44, 44, 44, 12, 12, 12
}
```

Definition at line 226 of file layer3.c.

7.3.4.21 unsigned char const sfb_32000_long[ ] [static]

Initial value:

```
 {
   4,  4,  4,  4,  4,  4,  6,  6,  8,  10,  12,
  16, 20, 24, 30, 38, 46, 56, 68, 84, 102,  26
}
```

Definition at line 157 of file layer3.c.

7.3.4.22 unsigned char const sfb_32000_mixed[ ] [static]

Initial value:

```
 {
     4,  4,  4,  4,  4,  4,  6,  6,
    4,  4,  4,  6,  6,  6,  8,  8,  8, 12,
              12, 12, 16, 16, 16, 20, 20, 20, 26, 26,
              26, 34, 34, 34, 42, 42, 42, 12, 12, 12
}
```

Definition at line 200 of file layer3.c.

7.3.4.23 unsigned char const sfb_32000_short[ ] [static]

Initial value:

```
 {
   4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  6,
   6,  6,  8,  8,  8, 12, 12, 12, 16, 16, 16, 20, 20,
  20, 26, 26, 26, 34, 34, 34, 42, 42, 42, 12, 12, 12
}
```

Definition at line 177 of file layer3.c.

7.3.4.24 unsigned char const sfb_44100_long[ ] [static]

Initial value:

```
 {
   4,  4,  4,  4,  4,  4,  6,  6,  8,   8,  10,
  12, 16, 20, 24, 28, 34, 42, 50, 54,  76, 158
}
```

Definition at line 151 of file layer3.c.

7.3.4.25 unsigned char const sfb_44100_mixed[ ] [static]

Initial value:

```
 {
     4,  4,  4,  4,  4,  4,  6,  6,
    4,  4,  4,  6,  6,  6,  8,  8,  8, 10,
              10, 10, 12, 12, 12, 14, 14, 14, 18, 18,
              18, 22, 22, 22, 30, 30, 30, 56, 56, 56
}
```

Definition at line 192 of file layer3.c.

7.3.4.26 unsigned char const sfb_44100_short[ ] [static]

Initial value:

```
 {
   4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  6,
   6,  6,  8,  8,  8, 10, 10, 10, 12, 12, 12, 14, 14,
  14, 18, 18, 18, 22, 22, 22, 30, 30, 30, 56, 56, 56
}
```

Definition at line 170 of file layer3.c.

7.3.4.27 unsigned char const sfb_48000_long[ ] [static]

Initial value:

```
 {
   4,  4,  4,  4,  4,  4,  6,  6,  6,   8,  10,
  12, 16, 18, 22, 28, 34, 40, 46, 54,  54, 192
}
```

MPEG-1 scalefactor band widths.

derived from Table B.8 of ISO/IEC 11172-3

Definition at line 145 of file layer3.c.

7.3.4.28 unsigned char const sfb_48000_mixed[ ] [static]

Initial value:

```
 {
     4,  4,  4,  4,  4,  4,  6,  6,
    4,  4,  4,  6,  6,  6,  6,  6,  6, 10,
              10, 10, 12, 12, 12, 14, 14, 14, 16, 16,
              16, 20, 20, 20, 26, 26, 26, 66, 66, 66
}
```

Definition at line 184 of file layer3.c.

7.3.4.29 unsigned char const sfb_48000_short[ ] [static]

Initial value:

```
 {
   4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  6,
   6,  6,  6,  6,  6, 10, 10, 10, 12, 12, 12, 14, 14,
  14, 16, 16, 16, 20, 20, 20, 26, 26, 26, 66, 66, 66
}
```

Definition at line 163 of file layer3.c.

7.3.4.30 unsigned char const sfb_8000_long[ ] [static]

Initial value:

```
 {
  12, 12, 12, 12, 12, 12, 16, 20, 24,  28,  32,
  40, 48, 56, 64, 76, 90,  2,  2,  2,   2,   2
}
```

Definition at line 278 of file layer3.c.

7.3.4.31 unsigned char const sfb_8000_mixed[ ] [static]

Initial value:

```
 {
    12, 12, 12,
    4,  4,  4,  8,  8,  8, 12, 12, 12, 16, 16, 16,
              20, 20, 20, 24, 24, 24, 28, 28, 28, 36, 36, 36,
               2,  2,  2,  2,  2,  2,  2,  2,  2, 26, 26, 26
}
```

Definition at line 299 of file layer3.c.

7.3.4.32 unsigned char const sfb_8000_short[ ] [static]

Initial value:

```
{
  8,  8,  8,  8,  8,  8,  8,  8,  8, 12, 12, 12, 16,
 16, 16, 20, 20, 20, 24, 24, 24, 28, 28, 28, 36, 36,
 36,  2,  2,  2,  2,  2,  2,  2,  2,  2, 26, 26, 26
}
```

Definition at line 287 of file layer3.c.

7.3.4.33 struct { ... } sfbwidth_table[9] [static]

Referenced by III_decode().

7.3.4.34 struct { ... } sflen_table[16] [static]

Referenced by III_scalefactors().

7.3.4.35 unsigned { ... } slen1 [inherited]

Definition at line 100 of file layer3.c.

7.3.4.36 unsigned { ... } slen2 [inherited]

Definition at line 101 of file layer3.c.

7.3.4.37 mad_fixed_t const window_l[36] [static]

Initial value:

```
{
 MAD_F(0x00b2aa3e) , MAD_F(0x0216a2a2) ,
 MAD_F(0x03768962) , MAD_F(0x04cfb0e2) ,
 MAD_F(0x061f78aa) , MAD_F(0x07635284) ,
 MAD_F(0x0898c779) , MAD_F(0x09bd7ca0) ,
 MAD_F(0x0acf37ad) , MAD_F(0x0bcbe352) ,
 MAD_F(0x0cb19346) , MAD_F(0x0d7e8807) ,

 MAD_F(0x0e313245) , MAD_F(0x0ec835e8) ,
 MAD_F(0x0f426cb5) , MAD_F(0x0f9ee890) ,
 MAD_F(0x0fdcf549) , MAD_F(0x0ffc19fd) ,
 MAD_F(0x0ffc19fd) , MAD_F(0x0fdcf549) ,
 MAD_F(0x0f9ee890) , MAD_F(0x0f426cb5) ,
 MAD_F(0x0ec835e8) , MAD_F(0x0e313245) ,

 MAD_F(0x0d7e8807) , MAD_F(0x0cb19346) ,
 MAD_F(0x0bcbe352) , MAD_F(0x0acf37ad) ,
 MAD_F(0x09bd7ca0) , MAD_F(0x0898c779) ,
```

```
  MAD_F(0x07635284) , MAD_F(0x061f78aa) ,
  MAD_F(0x04cfb0e2) , MAD_F(0x03768962) ,
  MAD_F(0x0216a2a2) , MAD_F(0x00b2aa3e) ,
}
```

windowing coefficients for long blocks.

derived from section 2.4.3.4.10.3 of ISO/IEC 11172-3

window_l[i] = sin((PI / 36) ∗ (i + 1/2))

Definition at line 406 of file layer3.c.

Referenced by III_imdct_l().

7.3.4.38 mad_fixed_t const window_s[12] [static]

Initial value:

```
 {
  MAD_F(0x0216a2a2) , MAD_F(0x061f78aa) ,
  MAD_F(0x09bd7ca0) , MAD_F(0x0cb19346) ,
  MAD_F(0x0ec835e8) , MAD_F(0x0fdcf549) ,
  MAD_F(0x0fdcf549) , MAD_F(0x0ec835e8) ,
  MAD_F(0x0cb19346) , MAD_F(0x09bd7ca0) ,
  MAD_F(0x061f78aa) , MAD_F(0x0216a2a2) ,
}
```

windowing coefficients for short blocks.

derived from section 2.4.3.4.10.3 of ISO/IEC 11172-3

window_s[i] = sin((PI / 12) ∗ (i + 1/2))

Definition at line 437 of file layer3.c.

Referenced by III_imdct_l(), and III_imdct_s().

7.4 Subband Synthesis Optimization

An optional optimization called here the Subband Synthesis Optimization (SSO) improves the performance of subband synthesis at the expense of accuracy.

Defines

- #define SHIFT(x) (x)

  FPM_DEFAULT without OPT_SSO will actually lose accuracy and performance.

- #define SHIFT(x) (x)

  FPM_DEFAULT without OPT_SSO will actually lose accuracy and performance.

- #define MUL(x, y) mad_f_mul((x), (y))

7.4.1 Detailed Description

An optional optimization called here the Subband Synthesis Optimization (SSO) improves the performance of subband synthesis at the expense of accuracy.

The idea is to simplify 32x32->64-bit multiplication to 32x32->32 such that extra scaling and rounding are not necessary. This often allows the compiler to use faster 32-bit multiply-accumulate instructions instead of explicit 64-bit multiply, shift, and add instructions.

SSO works like this: a full 32x32->64-bit multiply of two mad_fixed_t values requires the result to be right-shifted 28 bits to be properly scaled to the same fixed-point format. Right shifts can be applied at any time to either operand or to the result, so the optimization involves careful placement of these shifts to minimize the loss of accuracy.

First, a 14-bit shift is applied with rounding at compile-time to the D[] table of coefficients for the subband synthesis window. This only loses 2 bits of accuracy because the lower 12 bits are always zero. A second 12-bit shift occurs after the DCT calculation. This loses 12 bits of accuracy. Finally, a third 2-bit shift occurs just before the sample is saved in the PCM buffer. 14 + 12 + 2 == 28 bits.

7.4.2 Define Documentation

7.4.2.1 #define MUL(x, y) mad_f_mul((x), (y))

Definition at line 121 of file synth.c.

Referenced by dct32().

7.4.2.2 #define SHIFT(x) (x)

FPM_DEFAULT without OPT_SSO will actually lose accuracy and performance.

second SSO shift, with rounding.

Definition at line 540 of file synth.c.

7.4.2.3 #define SHIFT(x) (x)

FPM_DEFAULT without OPT_SSO will actually lose accuracy and performance.

second SSO shift, with rounding.

Definition at line 540 of file synth.c.

Referenced by dct32(), synth_full(), and synth_half().

Chapter 8

Data Structure Documentation

8.1 buffer Struct Reference

This is a private message structure.

Data Fields

- unsigned char const * start
- unsigned long length

8.1.1 Detailed Description

This is a private message structure.

A generic pointer to this structure is passed to each of the callback functions. Put here any data you need to access from within the callbacks.

Definition at line 72 of file minimad.c.

8.1.2 Field Documentation

8.1.2.1 unsigned char const* buffer::start

Definition at line 73 of file minimad.c.

Referenced by decode(), error(), and input().

8.1.2.2 unsigned long buffer::length

Definition at line 74 of file minimad.c.

Referenced by decode(), and input().

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

- minimad.c

8.2 fixedfloat Struct Reference

table for requantization.

Data Fields

- unsigned long mantissa: 27
- unsigned short exponent: 5

8.2.1 Detailed Description

table for requantization.

rq_table[x].mantissa * 2^(rq_table[x].exponent) = x^(4/3)

Definition at line 338 of file layer3.c.

8.2.2 Field Documentation

8.2.2.1 unsigned long fixedfloat::mantissa

Definition at line 339 of file layer3.c.

Referenced by III_requantize().

8.2.2.2 unsigned short fixedfloat::exponent

Definition at line 340 of file layer3.c.

Referenced by III_requantize().

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

- layer3.c

8.3 huffpair Union Reference

`#include <huffman.h>`

Data Fields

- struct {
  unsigned short final: 1
  unsigned short bits: 3
  unsigned short offset: 12
  } ptr

- struct {
  unsigned short final: 1
  unsigned short hlen: 3
  unsigned short x: 4
  unsigned short y: 4
  } value

8.3.1 Detailed Description

Definition at line 45 of file huffman.h.

8.3.2 Field Documentation

8.3.2.1 unsigned short huffpair::final

Definition at line 47 of file huffman.h.

Referenced by III_huffdecode().

8.3.2.2 unsigned short huffpair::bits

Definition at line 48 of file huffman.h.

Referenced by III_huffdecode().

8.3.2.3 unsigned short huffpair::offset

Definition at line 49 of file huffman.h.

Referenced by III_huffdecode().

8.3.2.4 struct { ... } huffpair::ptr

Referenced by III_huffdecode().

8.3.2.5 unsigned short huffpair::hlen

Definition at line 53 of file huffman.h.

Referenced by III_huffdecode().

8.3.2.6 unsigned short huffpair::x

Definition at line 54 of file huffman.h.

Referenced by III_huffdecode().

8.3.2.7 unsigned short huffpair::y

Definition at line 55 of file huffman.h.

Referenced by III_huffdecode().

8.3.2.8 struct { ... } huffpair::value

Referenced by III_huffdecode().

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

- huffman.h

8.4 huffquad Union Reference

`#include <huffman.h>`

Data Fields

- struct {
  unsigned short final: 1
  unsigned short bits: 3
  unsigned short offset: 12
  } ptr

- struct {
  unsigned short final: 1
  unsigned short hlen: 3
  unsigned short v: 1
  unsigned short w: 1
  unsigned short x: 1
  unsigned short y: 1
  } value

8.4.1 Detailed Description

Definition at line 28 of file huffman.h.

8.4.2 Field Documentation

8.4.2.1 unsigned short huffquad::final

Definition at line 30 of file huffman.h.

Referenced by III_huffdecode().

8.4.2.2 unsigned short huffquad::bits

Definition at line 31 of file huffman.h.

Referenced by III_huffdecode().

8.4.2.3 unsigned short huffquad::offset

Definition at line 32 of file huffman.h.

Referenced by III_huffdecode().

8.4.2.4 struct { ... } huffquad::ptr

Referenced by III_huffdecode().

8.4.2.5 unsigned short huffquad::hlen

Definition at line 36 of file huffman.h.

Referenced by III_huffdecode().

8.4.2.6 unsigned short huffquad::v

Definition at line 37 of file huffman.h.

Referenced by III_huffdecode().

8.4.2.7 unsigned short huffquad::w

Definition at line 38 of file huffman.h.

Referenced by III_huffdecode(), and III_reorder().

8.4.2.8 unsigned short huffquad::x

Definition at line 39 of file huffman.h.

Referenced by III_huffdecode().

8.4.2.9 unsigned short huffquad::y

Definition at line 40 of file huffman.h.

Referenced by III_huffdecode().

8.4.2.10 struct { ... } huffquad::value

Referenced by III_huffdecode().

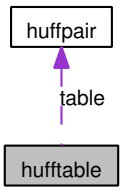
The documentation for this union was generated from the following file:

- huffman.h

8.5 hufftable Struct Reference

`#include <huffman.h>`

Collaboration diagram for hufftable:



Data Fields

- union huffpair const * table
- unsigned short linbits
- unsigned short startbits

8.5.1 Detailed Description

Definition at line 60 of file huffman.h.

8.5.2 Field Documentation

8.5.2.1 union huffpair const* hufftable::table [write]

Definition at line 61 of file huffman.h.

Referenced by III_huffdecode().

8.5.2.2 unsigned short hufftable::linbits

Definition at line 62 of file huffman.h.

Referenced by III_huffdecode().

8.5.2.3 unsigned short hufftable::startbits

Definition at line 63 of file huffman.h.

Referenced by III_huffdecode().

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

- huffman.h

8.6 mad_bitptr Struct Reference

`#include <bit.h>`

Data Fields

- unsigned char const * byte
- unsigned short cache
- unsigned short left

8.6.1 Detailed Description

Definition at line 29 of file bit.h.

8.6.2 Field Documentation

8.6.2.1 unsigned char const* mad_bitptr::byte

Definition at line 30 of file bit.h.

Referenced by mad_bit_init(), mad_bit_length(), mad_bit_nextbyte(), mad_bit_read(), and mad_bit_skip().

8.6.2.2 unsigned short mad_bitptr::cache

Definition at line 31 of file bit.h.

Referenced by mad_bit_init(), mad_bit_read(), and mad_bit_skip().

8.6.2.3 unsigned short mad_bitptr::left

Definition at line 32 of file bit.h.

Referenced by mad_bit_init(), mad_bit_length(), mad_bit_nextbyte(), mad_bit_read(), and mad_bit_skip().

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

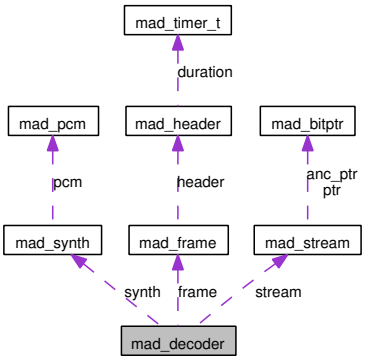
- bit.h

8.7 mad_decoder Struct Reference

This holds all information about how you want your stream decoded, such as input/output functions, error handling functions, etc.

`#include <decoder.h>`

Collaboration diagram for mad_decoder:



Data Fields

- enum mad_decoder_mode mode
- int options
- struct {
  long pid
  int in
  int out
  } async

- struct {
  struct mad_stream stream
  struct mad_frame frame
  struct mad_synth synth
  } sync

- void * cb_data
- input_callback input_func
- header_callback header_func
- filter_callback filter_func
- output_callback output_func
- error_callback error_func
- message_callback message_func

8.7.1 Detailed Description

This holds all information about how you want your stream decoded, such as input/output functions, error handling functions, etc.

Definition at line 50 of file decoder.h.

8.7.2 Field Documentation

8.7.2.1 enum mad_decoder_mode mad_decoder::mode

Definition at line 51 of file decoder.h.

Referenced by mad_decoder_finish(), mad_decoder_init(), mad_decoder_message(), mad_decoder_run(), and run_sync().

8.7.2.2 int mad_decoder::options

Definition at line 53 of file decoder.h.

Referenced by mad_decoder_init(), and run_sync().

8.7.2.3 long mad_decoder::pid

Definition at line 56 of file decoder.h.

Referenced by mad_decoder_finish(), and mad_decoder_init().

8.7.2.4 int mad_decoder::in

Definition at line 57 of file decoder.h.

Referenced by mad_decoder_finish(), mad_decoder_init(), and mad_decoder_message().

8.7.2.5 int mad_decoder::out

Definition at line 58 of file decoder.h.

Referenced by mad_decoder_finish(), mad_decoder_init(), and mad_decoder_message().

8.7.2.6 struct { ... } mad_decoder::async

Referenced by mad_decoder_finish(), mad_decoder_init(), and mad_decoder_message().

8.7.2.7 struct mad_stream mad_decoder::stream [read]

Definition at line 62 of file decoder.h.

Referenced by run_sync().

8.7.2.8 struct mad_frame mad_decoder::frame [read]

Definition at line 63 of file decoder.h.

Referenced by run_sync().

8.7.2.9 struct mad_synth mad_decoder::synth [read]

Definition at line 64 of file decoder.h.

Referenced by run_sync().

8.7.2.10 struct { ... } * mad_decoder::sync

Referenced by mad_decoder_init(), mad_decoder_run(), and run_sync().

8.7.2.11 void* mad_decoder::cb_data

Definition at line 67 of file decoder.h.

Referenced by mad_decoder_init(), and run_sync().

8.7.2.12 input_callback mad_decoder::input_func

Definition at line 69 of file decoder.h.

Referenced by mad_decoder_init(), and run_sync().

8.7.2.13 header_callback mad_decoder::header_func

Definition at line 70 of file decoder.h.

Referenced by mad_decoder_init(), and run_sync().

8.7.2.14 filter_callback mad_decoder::filter_func

Definition at line 71 of file decoder.h.

Referenced by mad_decoder_init(), and run_sync().

8.7.2.15 output_callback mad_decoder::output_func

Definition at line 72 of file decoder.h.

Referenced by mad_decoder_init(), and run_sync().

8.7.2.16 error_callback mad_decoder::error_func

Definition at line 73 of file decoder.h.

Referenced by mad_decoder_init(), and run_sync().

8.7.2.17 message_callback mad_decoder::message_func

Definition at line 74 of file decoder.h.

Referenced by mad_decoder_init().

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

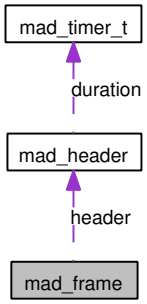
- decoder.h

8.8 mad_frame Struct Reference

MPEG unit of encoding, and associated context.

`#include <frame.h>`

Collaboration diagram for mad_frame:



Data Fields

- struct mad_header header
  MPEG audio header.
- int options
  decoding options (from stream)
- mad_fixed_t sbsample [2][36][32]
  synthesis subband filter samples
- mad_fixed_t(∗ overlap)[2][32][18]
  Layer III block overlap data.

8.8.1 Detailed Description

MPEG unit of encoding, and associated context.

Definition at line 76 of file frame.h.

8.8.2 Field Documentation

8.8.2.1 struct mad_header mad_frame::header [read]

MPEG audio header.

Definition at line 77 of file frame.h.

Referenced by III_decode(), mad_frame_decode(), mad_frame_finish(), mad_frame_-init(), mad_layer_I(), mad_layer_II(), mad_layer_III(), mad_synth_frame(), and run_-sync().

8.8.2.2 int mad_frame::options

decoding options (from stream)

Definition at line 79 of file frame.h.

Referenced by mad_frame_decode(), mad_frame_init(), mad_layer_I(), mad_layer_-II(), mad_layer_III(), and mad_synth_frame().

8.8.2.3 mad_fixed_t mad_frame::sbsample[2][36][32]

synthesis subband filter samples

Definition at line 81 of file frame.h.

Referenced by III_decode(), mad_frame_mute(), mad_layer_I(), mad_layer_II(), synth_full(), and synth_half().

8.8.2.4 mad_fixed_t(∗ mad_frame::overlap)[2][32][18]

Layer III block overlap data.

Definition at line 82 of file frame.h.

Referenced by III_decode(), mad_frame_finish(), mad_frame_init(), mad_frame_-mute(), and mad_layer_III().

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

- frame.h

8.9 mad_header Struct Reference

MPEG frame header information.

`#include <frame.h>`

Collaboration diagram for mad_header:

Data Fields

- enum mad_layer layer

  audio layer (1, 2, or 3)

- enum mad_mode mode

  channel mode (see above)

- int mode_extension

  additional mode info

- enum mad_emphasis emphasis

  de-emphasis to use (see above)

- unsigned long bitrate

  stream bitrate (bps)

- unsigned int samplerate

  sampling frequency (Hz)

- unsigned short crc_check

  frame CRC accumulator

- unsigned short crc_target

  final target CRC checksum

- int flags

  flags (see below)

- int private_bits

  private bits (see below)

- mad_timer_t duration

  audio playing time of frame

8.9.1 Detailed Description

MPEG frame header information.

Definition at line 55 of file frame.h.

8.9.2 Field Documentation

8.9.2.1 enum mad_layer mad_header::layer

audio layer (1, 2, or 3)

Definition at line 56 of file frame.h.

Referenced by decode_header(), free_bitrate(), mad_frame_decode(), mad_header_-decode(), and mad_header_init().

8.9.2.2 enum mad_mode mad_header::mode

channel mode (see above)

Definition at line 57 of file frame.h.

Referenced by decode_header(), III_decode(), mad_header_init(), mad_layer_I(), and mad_layer_II().

8.9.2.3 int mad_header::mode_extension

additional mode info

Definition at line 58 of file frame.h.

Referenced by decode_header(), III_decode(), III_stereo(), mad_header_init(), mad_-layer_I(), and mad_layer_II().

8.9.2.4 enum mad_emphasis mad_header::emphasis

de-emphasis to use (see above)

Definition at line 59 of file frame.h.

Referenced by decode_header(), and mad_header_init().

8.9.2.5 unsigned long mad_header::bitrate

stream bitrate (bps)

Definition at line 61 of file frame.h.

Referenced by decode_header(), mad_header_decode(), mad_header_init(), and mad_-layer_II().

8.9.2.6 unsigned int mad_header::samplerate

sampling frequency (Hz)

Definition at line 62 of file frame.h.

Referenced by decode_header(), free_bitrate(), III_decode(), mad_header_decode(), mad_header_init(), mad_layer_II(), and mad_synth_frame().

8.9.2.7 unsigned short mad_header::crc_check

frame CRC accumulator

Definition at line 64 of file frame.h.

Referenced by decode_header(), mad_header_init(), mad_layer_I(), mad_layer_II(), and mad_layer_III().

8.9.2.8 unsigned short mad_header::crc_target

final target CRC checksum

Definition at line 65 of file frame.h.

Referenced by decode_header(), mad_header_init(), mad_layer_I(), mad_layer_II(), and mad_layer_III().

8.9.2.9 int mad_header::flags

flags (see below)

Definition at line 67 of file frame.h.

Referenced by decode_header(), free_bitrate(), III_decode(), III_stereo(), mad_frame_decode(), mad_header_decode(), mad_header_init(), mad_layer_I(), mad_layer_II(), and mad_layer_III().

8.9.2.10 int mad_header::private_bits

private bits (see below)

Definition at line 68 of file frame.h.

Referenced by decode_header(), mad_header_init(), and mad_layer_III().

8.9.2.11 mad_timer_t mad_header::duration

audio playing time of frame

Definition at line 70 of file frame.h.

Referenced by mad_header_decode(), and mad_header_init().

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

- frame.h

8.10 mad_pcm Struct Reference

`#include <synth.h>`

Data Fields

- unsigned int samplerate
  sampling frequency (Hz)
- unsigned short channels
  number of channels
- unsigned short length
  number of samples per channel
- mad_fixed_t samples [2][1152]
  PCM output samples [ch][sample].

8.10.1 Detailed Description

Definition at line 30 of file synth.h.

8.10.2 Field Documentation

8.10.2.1 unsigned int mad_pcm::samplerate

sampling frequency (Hz)

Definition at line 31 of file synth.h.

Referenced by mad_synth_frame(), and mad_synth_init().

8.10.2.2 unsigned short mad_pcm::channels

number of channels

Definition at line 32 of file synth.h.

Referenced by mad_synth_frame(), mad_synth_init(), and output().

8.10.2.3 unsigned short mad_pcm::length

number of samples per channel

Definition at line 33 of file synth.h.

Referenced by mad_synth_frame(), mad_synth_init(), and output().

8.10.2.4 mad_fixed_t mad_pcm::samples[2][1152]

PCM output samples [ch][sample].

Definition at line 34 of file synth.h.

Referenced by output(), synth_full(), and synth_half().

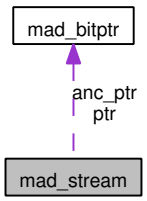
The documentation for this struct was generated from the following file:

- synth.h

8.11 mad_stream Struct Reference

`#include <stream.h>`

Collaboration diagram for mad_stream:



Data Fields

- unsigned char const * buffer
  input bitstream buffer
- unsigned char const * bufend
  end of buffer
- unsigned long skiplen
  bytes to skip before next frame
- int sync
  stream sync found
- unsigned long freerate
  free bitrate (fixed)
- unsigned char const * this_frame
  start of current frame
- unsigned char const * next_frame
  start of next frame
- struct mad_bitptr ptr
  current processing bit pointer
- struct mad_bitptr anc_ptr
  ancillary bits pointer
- unsigned int anc_bitlen
  number of ancillary bits

- unsigned char(* main_data)[MAD_BUFFER_MDLEN]
  Layer III main_data().
- unsigned int md_len
  bytes in main_data
- int options
  decoding options (see below)
- enum mad_error error
  error code (see above)

8.11.1 Detailed Description

Definition at line 65 of file stream.h.

8.11.2 Field Documentation

8.11.2.1 unsigned char const* mad_stream::buffer

input bitstream buffer

Definition at line 66 of file stream.h.

Referenced by mad_stream_buffer(), and mad_stream_init().

8.11.2.2 unsigned char const* mad_stream::bufend

end of buffer

Definition at line 67 of file stream.h.

Referenced by mad_header_decode(), mad_stream_buffer(), mad_stream_init(), and mad_stream_sync().

8.11.2.3 unsigned long mad_stream::skiplen

bytes to skip before next frame

Definition at line 68 of file stream.h.

Referenced by mad_header_decode(), mad_stream_init(), and mad_stream_skip().

8.11.2.4 int mad_stream::sync

stream sync found

Definition at line 70 of file stream.h.

Referenced by mad_header_decode(), mad_stream_buffer(), and mad_stream_init().

8.11.2.5 unsigned long mad_stream::freerate

free bitrate (fixed)

Definition at line 71 of file stream.h.

Referenced by free_bitrate(), mad_header_decode(), and mad_stream_init().

8.11.2.6 unsigned char const* mad_stream::this_frame

start of current frame

Definition at line 73 of file stream.h.

Referenced by error(), free_bitrate(), mad_frame_decode(), mad_header_decode(), mad_stream_buffer(), and mad_stream_init().

8.11.2.7 unsigned char const* mad_stream::next_frame

start of next frame

Definition at line 74 of file stream.h.

Referenced by mad_frame_decode(), mad_header_decode(), mad_layer_III(), mad_stream_buffer(), and mad_stream_init().

8.11.2.8 struct mad_bitptr mad_stream::ptr [read]

current processing bit pointer

Definition at line 75 of file stream.h.

Referenced by decode_header(), free_bitrate(), mad_frame_decode(), mad_header_decode(), mad_layer_I(), mad_layer_II(), mad_layer_III(), mad_stream_buffer(), mad_stream_finish(), mad_stream_init(), and mad_stream_sync().

8.11.2.9 struct mad_bitptr mad_stream::anc_ptr [read]

ancillary bits pointer

Definition at line 77 of file stream.h.

Referenced by mad_frame_decode(), mad_layer_III(), mad_stream_finish(), and mad_stream_init().

8.11.2.10 unsigned int mad_stream::anc_bitlen

number of ancillary bits

Definition at line 78 of file stream.h.

Referenced by mad_frame_decode(), mad_layer_III(), and mad_stream_init().

8.11.2.11 unsigned char(* mad_stream::main_data)[MAD_BUFFER_MDLEN]

Layer III main_data().

Definition at line 80 of file stream.h.

Referenced by mad_layer_III(), mad_stream_finish(), and mad_stream_init().

8.11.2.12 unsigned int mad_stream::md_len

bytes in main_data

Definition at line 82 of file stream.h.

Referenced by mad_layer_III(), and mad_stream_init().

8.11.2.13 int mad_stream::options

decoding options (see below)

Definition at line 84 of file stream.h.

Referenced by mad_frame_decode(), and mad_stream_init().

8.11.2.14 enum mad_error mad_stream::error

error code (see above)

Definition at line 85 of file stream.h.

Referenced by decode_header(), error(), error_default(), free_bitrate(), mad_frame_decode(), mad_header_decode(), mad_layer_I(), mad_layer_II(), mad_layer_III(), mad_stream_errorstr(), mad_stream_init(), and run_sync().

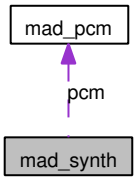
The documentation for this struct was generated from the following file:

- stream.h

8.12 mad_synth Struct Reference

`#include <synth.h>`

Collaboration diagram for mad_synth:



Data Fields

- mad_fixed_t filter [2][2][2][16][8]
  polyphase filterbank outputs
- unsigned int phase
  current processing phase
- struct mad_pcm pcm
  PCM output.

8.12.1 Detailed Description

Definition at line 37 of file synth.h.

8.12.2 Field Documentation

8.12.2.1 mad_fixed_t mad_synth::filter[2][2][2][16][8]

polyphase filterbank outputs

[ch][eo][peo][s][v]

Definition at line 38 of file synth.h.

Referenced by mad_synth_mute(), synth_full(), and synth_half().

8.12.2.2 unsigned int mad_synth::phase

current processing phase

Definition at line 41 of file synth.h.

Referenced by mad_synth_frame(), mad_synth_init(), synth_full(), and synth_half().

8.12.2.3 struct mad_pcm mad_synth::pcm [read]

PCM output.

Definition at line 43 of file synth.h.

Referenced by mad_synth_frame(), mad_synth_init(), run_sync(), synth_full(), and synth_half().

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

- synth.h

8.13 mad_timer_t Struct Reference

`#include <timer.h>`

Data Fields

- signed long seconds
  whole seconds
- unsigned long fraction
  1/MAD_TIMER_RESOLUTION seconds

8.13.1 Detailed Description

Definition at line 27 of file timer.h.

8.13.2 Field Documentation

8.13.2.1 signed long mad_timer_t::seconds

whole seconds

Definition at line 28 of file timer.h.

Referenced by mad_timer_abs(), mad_timer_add(), mad_timer_compare(), mad_timer_count(), mad_timer_negate(), mad_timer_set(), mad_timer_string(), and reduce_timer().

8.13.2.2 unsigned long mad_timer_t::fraction

1/MAD_TIMER_RESOLUTION seconds

Definition at line 29 of file timer.h.

Referenced by mad_timer_add(), mad_timer_compare(), mad_timer_count(), mad_timer_fraction(), mad_timer_negate(), mad_timer_set(), mad_timer_string(), and reduce_timer().

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

- timer.h

8.14 quantclass Struct Reference

quantization class table

Data Fields

- unsigned short nlevels
- unsigned char group
- unsigned char bits
- mad_fixed_t C
- mad_fixed_t D

8.14.1 Detailed Description

quantization class table

Definition at line 278 of file layer12.c.

8.14.2 Field Documentation

8.14.2.1 unsigned short quantclass::nlevels

Definition at line 279 of file layer12.c.

Referenced by II_samples().

8.14.2.2 unsigned char quantclass::group

Definition at line 280 of file layer12.c.

Referenced by II_samples().

8.14.2.3 unsigned char quantclass::bits

Definition at line 281 of file layer12.c.

Referenced by II_samples().

8.14.2.4 mad_fixed_t quantclass::C

Definition at line 282 of file layer12.c.

Referenced by II_samples().

8.14.2.5 mad_fixed_t quantclass::D

Definition at line 283 of file layer12.c.

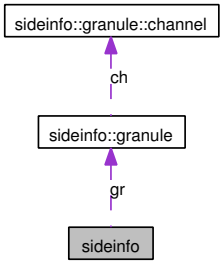
Referenced by II_samples().

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

- layer12.c

8.15 sideinfo Struct Reference

Collaboration diagram for sideinfo:



Data Fields

- unsigned int main_data_begin
- unsigned int private_bits
- unsigned char scfsi [2]
- struct sideinfo::granule gr [2]

Data Structures

- struct granule

8.15.1 Detailed Description

Definition at line 67 of file layer3.c.

8.15.2 Field Documentation

8.15.2.1 unsigned int sideinfo::main_data_begin

Definition at line 68 of file layer3.c.

Referenced by III_sideinfo(), and mad_layer_III().

8.15.2.2 unsigned int sideinfo::private_bits

Definition at line 69 of file layer3.c.

Referenced by III_sideinfo(), and mad_layer_III().

8.15.2.3 unsigned char sideinfo::scfsi[2]

Definition at line 71 of file layer3.c.

Referenced by III_decode(), and III_sideinfo().

8.15.2.4 struct sideinfo::granule sideinfo::gr[2]

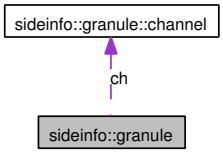
Referenced by III_decode(), and III_sideinfo().

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

- layer3.c

8.16 sideinfo::granule Struct Reference

Collaboration diagram for sideinfo::granule:



Data Fields

- struct sideinfo::granule::channel ch [2]

Data Structures

- struct channel

8.16.1 Detailed Description

Definition at line 73 of file layer3.c.

8.16.2 Field Documentation

8.16.2.1 struct sideinfo::granule::channel sideinfo::granule::ch[2]

Referenced by III_decode(), and III_sideinfo().

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

- layer3.c

8.17 sideinfo::granule::channel Struct Reference

Data Fields

- unsigned short part2_3_length
- unsigned short big_values
- unsigned short global_gain
- unsigned short scalefac_compress
- unsigned char flags
- unsigned char block_type
- unsigned char table_select [3]
- unsigned char subblock_gain [3]
- unsigned char region0_count
- unsigned char region1_count
- unsigned char scalefac [39]

8.17.1 Detailed Description

Definition at line 74 of file layer3.c.

8.17.2 Field Documentation

8.17.2.1 unsigned short sideinfo::granule::channel::part2_3_length

Definition at line 76 of file layer3.c.

Referenced by III_sideinfo().

8.17.2.2 unsigned short sideinfo::granule::channel::big_values

Definition at line 77 of file layer3.c.

8.17.2.3 unsigned short sideinfo::granule::channel::global_gain

Definition at line 78 of file layer3.c.

8.17.2.4 unsigned short sideinfo::granule::channel::scalefac_compress

Definition at line 79 of file layer3.c.

8.17.2.5 unsigned char sideinfo::granule::channel::flags

Definition at line 81 of file layer3.c.

8.17.2.6 unsigned char sideinfo::granule::channel::block_type

Definition at line 82 of file layer3.c.

8.17.2.7 unsigned char sideinfo::granule::channel::table_select[3]

Definition at line 83 of file layer3.c.

8.17.2.8 unsigned char sideinfo::granule::channel::subblock_gain[3]

Definition at line 84 of file layer3.c.

8.17.2.9 unsigned char sideinfo::granule::channel::region0_count

Definition at line 85 of file layer3.c.

8.17.2.10 unsigned char sideinfo::granule::channel::region1_count

Definition at line 86 of file layer3.c.

8.17.2.11 unsigned char sideinfo::granule::channel::scalefac[39]

Definition at line 89 of file layer3.c.

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

- layer3.c

Chapter 9

File Documentation

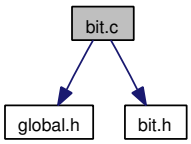
9.1 bit.c File Reference

Manipulation methods for the mad_bitptr structure, used in various low-level routines.

```
#include "global.h"
```

```
#include "bit.h"
```

Include dependency graph for bit.c:



Defines

- #define CHAR_BIT 8
- #define CRC_POLY 0x8005

Functions

- void mad_bit_init (struct mad_bitptr *bitptr, unsigned char const *byte)

  Initializes bit pointer struct.

- unsigned int mad_bit_length (struct mad_bitptr const *begin, struct mad_bitptr const *end)

  Returns number of bits between start and end points.

- unsigned char const * mad_bit_nextbyte (struct mad_bitptr const *bitptr)

Returns pointer to next unprocessed byte.

- void mad_bit_skip (struct mad_bitptr *bitptr, unsigned int len)

  Advances bit pointer.

- unsigned long mad_bit_read (struct mad_bitptr *bitptr, unsigned int len)

  Reads an arbitrary number of bits and return their UIMSBF value.

- unsigned short mad_bit_crc (struct mad_bitptr bitptr, unsigned int len, unsigned short init)

  Computes CRC-check word.

Variables

- static unsigned short const crc_table [256]

  This is the lookup table for computing the CRC-check word.

9.1.1 Detailed Description

Manipulation methods for the mad_bitptr structure, used in various low-level routines.

Definition in file bit.c.

9.1.2 Define Documentation

9.1.2.1 #define CHAR_BIT 8

Definition at line 35 of file bit.c.

Referenced by III_huffdecode(), III_requantize(), mad_bit_init(), mad_bit_length(), mad_bit_nextbyte(), mad_bit_read(), mad_bit_skip(), and mad_layer_III().

9.1.2.2 #define CRC_POLY 0x8005

Definition at line 86 of file bit.c.

Referenced by mad_bit_crc().

9.1.3 Function Documentation

9.1.3.1 unsigned short mad_bit_crc (struct mad_bitptr *bitptr*, unsigned int *len*, unsigned short *init*)

Computes CRC-check word.

Definition at line 194 of file bit.c.

References CRC_POLY, crc_table, and mad_bit_read().

Referenced by decode_header(), mad_layer_I(), mad_layer_II(), and mad_layer_III().

Here is the call graph for this function:

9.1.3.2 void mad_bit_init (struct mad_bitptr * *bitptr*, unsigned char const * *byte*)

Initializes bit pointer struct.

Definition at line 91 of file bit.c.

References mad_bitptr::byte, mad_bitptr::cache, CHAR_BIT, and mad_bitptr::left.

Referenced by mad_frame_decode(), mad_header_decode(), mad_layer_III(), mad_stream_buffer(), mad_stream_init(), and mad_stream_sync().

9.1.3.3 unsigned int mad_bit_length (struct mad_bitptr const * *begin*, struct mad_bitptr const * *end*)

Returns number of bits between start and end points.

Definition at line 101 of file bit.c.

References mad_bitptr::byte, CHAR_BIT, and mad_bitptr::left.

Referenced by III_scalefactors(), III_scalefactors_lsf(), mad_frame_decode(), and mad_layer_II().

9.1.3.4 unsigned char const* mad_bit_nextbyte (struct mad_bitptr const * *bitptr*)

Returns pointer to next unprocessed byte.

Definition at line 111 of file bit.c.

References mad_bitptr::byte, CHAR_BIT, and mad_bitptr::left.

Referenced by free_bitrate(), mad_header_decode(), mad_layer_III(), and mad_stream_sync().

9.1.3.5 unsigned long mad_bit_read (struct mad_bitptr * *bitptr*, unsigned int *len*)

Reads an arbitrary number of bits and return their UIMSBF value.

Definition at line 136 of file bit.c.

References mad_bitptr::byte, mad_bitptr::cache, CHAR_BIT, and mad_bitptr::left.

Referenced by decode_header(), I_sample(), II_samples(), III_huffdecode(), III_scalefactors(), III_scalefactors_lsf(), III_sideinfo(), mad_bit_crc(), mad_layer_I(), mad_layer_II(), and mad_layer_III().

9.1.3.6 void mad_bit_skip (struct mad_bitptr * *bitptr*, unsigned int *len*)

Advances bit pointer.

Definition at line 119 of file bit.c.

References mad_bitptr::byte, mad_bitptr::cache, CHAR_BIT, and mad_bitptr::left.

Referenced by decode_header(), free_bitrate(), III_huffdecode(), and mad_layer_III().

9.1.4 Variable Documentation

9.1.4.1 unsigned short const crc_table[256] [static]

This is the lookup table for computing the CRC-check word.

As described in section 2.4.3.1 and depicted in Figure A.9 of ISO/IEC 11172-3, the generator polynomial is:

G(X) = X^16 + X^15 + X^2 + 1

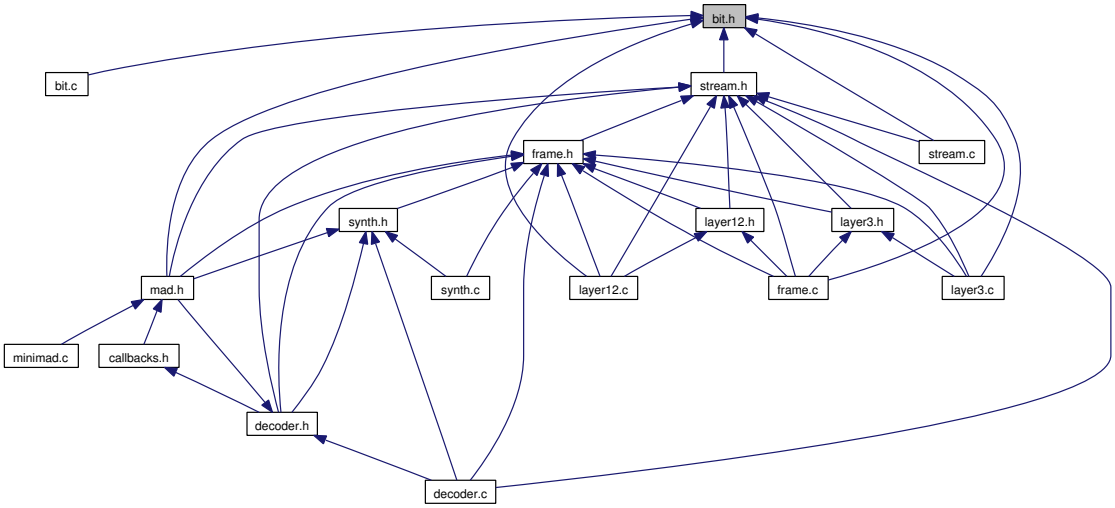
Definition at line 48 of file bit.c.

Referenced by mad_bit_crc().

9.2 bit.h File Reference

Definition of the mad_bit_ptr structure and methods, used in various low-level routines.

This graph shows which files directly or indirectly include this file:



Data Structures

- struct mad_bitptr

Defines

- #define mad_bit_finish(bitptr)
  nothing
- #define mad_bit_bitsleft(bitptr) ((bitptr) → left)

Functions

- void mad_bit_init (struct mad_bitptr *, unsigned char const *)
  Initializes bit pointer struct.
- unsigned int mad_bit_length (struct mad_bitptr const *, struct mad_bitptr const *)
  Returns number of bits between start and end points.
- unsigned char const * mad_bit_nextbyte (struct mad_bitptr const *)
  Returns pointer to next unprocessed byte.

- void mad_bit_skip (struct mad_bitptr *, unsigned int)
  Advances bit pointer.
- unsigned long mad_bit_read (struct mad_bitptr *, unsigned int)
  Reads an arbitrary number of bits and return their UIMSBF value.
- void mad_bit_write (struct mad_bitptr *, unsigned int, unsigned long)
- unsigned short mad_bit_crc (struct mad_bitptr, unsigned int, unsigned short)
  Computes CRC-check word.

9.2.1 Detailed Description

Definition of the mad_bit_ptr structure and methods, used in various low-level routines.

Definition in file bit.h.

9.2.2 Define Documentation

9.2.2.1 #define mad_bit_bitsleft(bitptr) ((bitptr) → left)

Definition at line 42 of file bit.h.

Referenced by III_huffdecode().

9.2.2.2 #define mad_bit_finish(bitptr)

nothing

Definition at line 37 of file bit.h.

Referenced by mad_frame_decode(), mad_layer_III(), and mad_stream_finish().

9.2.3 Function Documentation

9.2.3.1 unsigned short mad_bit_crc (struct *mad_bitptr*, unsigned *int*, unsigned *short*)

Computes CRC-check word.

Definition at line 194 of file bit.c.

References CRC_POLY, crc_table, and mad_bit_read().

Referenced by decode_header(), mad_layer_I(), mad_layer_II(), and mad_layer_III().

Here is the call graph for this function:

9.2.3.2 void mad_bit_init (struct mad_bitptr *, unsigned char const *)

Initializes bit pointer struct.

Definition at line 91 of file bit.c.

References mad_bitptr::byte, mad_bitptr::cache, CHAR_BIT, and mad_bitptr::left.

Referenced by mad_frame_decode(), mad_header_decode(), mad_layer_III(), mad_stream_buffer(), mad_stream_init(), and mad_stream_sync().

9.2.3.3 unsigned int mad_bit_length (struct mad_bitptr const *, struct mad_bitptr const *)

Returns number of bits between start and end points.

Definition at line 101 of file bit.c.

References mad_bitptr::byte, CHAR_BIT, and mad_bitptr::left.

Referenced by III_scalefactors(), III_scalefactors_lsf(), mad_frame_decode(), and mad_layer_II().

9.2.3.4 unsigned char const* mad_bit_nextbyte (struct mad_bitptr const *)

Returns pointer to next unprocessed byte.

Definition at line 111 of file bit.c.

References mad_bitptr::byte, CHAR_BIT, and mad_bitptr::left.

Referenced by free_bitrate(), mad_header_decode(), mad_layer_III(), and mad_stream_sync().

9.2.3.5 unsigned long mad_bit_read (struct mad_bitptr *, unsigned *int*)

Reads an arbitrary number of bits and return their UIMSBF value.

Definition at line 136 of file bit.c.

References mad_bitptr::byte, mad_bitptr::cache, CHAR_BIT, and mad_bitptr::left.

Referenced by decode_header(), I_sample(), II_samples(), III_huffdecode(), III_scalefactors(), III_scalefactors_lsf(), III_sideinfo(), mad_bit_crc(), mad_layer_I(), mad_layer_II(), and mad_layer_III().

9.2.3.6 void mad_bit_skip (struct mad_bitptr *, unsigned *int*)

Advances bit pointer.

Definition at line 119 of file bit.c.

References mad_bitptr::byte, mad_bitptr::cache, CHAR_BIT, and mad_bitptr::left.

Referenced by decode_header(), free_bitrate(), III_huffdecode(), and mad_layer_III().

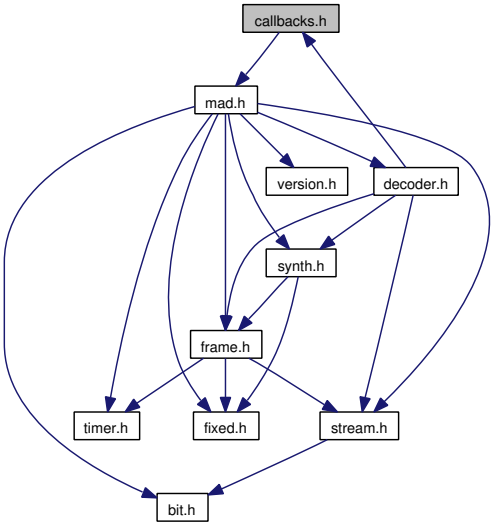
9.2.3.7 void mad_bit_write (struct mad_bitptr *, unsigned *int*, unsigned *long*)

9.3 callbacks.h File Reference

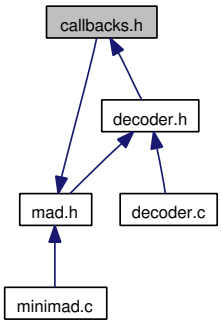
Callback functions definitions for use in mad_decoder struct.

```
#include "mad.h"
```

Include dependency graph for callbacks.h:



This graph shows which files directly or indirectly include this file:



Variables

- enum mad_flow(∗ input_callback)(void ∗user_data, struct mad_stream ∗stream)

  The purpose of this callback is to (re)fill the stream buffer which is to be decoded.

- enum mad_flow(∗ header_callback)(void ∗user_data, struct mad_header const ∗header)

The function called after MPEG headers have been decoded .

- enum mad_flow(∗ filter_callback)(void ∗user_data, struct mad_stream const ∗stream, struct mad_frame ∗)

  The filter callback function is called after decoding a frame, but before synthesis.

- enum mad_flow(∗ output_callback)(void ∗user_data, struct mad_header const ∗header, struct mad_pcm ∗pcm)

  The purpose of this callback is to output (or play) the decoded PCM audio.

- enum mad_flow(∗ error_callback)(void ∗user_data, struct mad_stream ∗stream, struct mad_frame ∗frame)

  The error callback function is called whenever a decoding error occurs.

- enum mad_flow(∗ message_callback)(void ∗user_data, void ∗message, unsigned int ∗size)

  The message callback function is only used with MAD_DECODER_MODE_ASYNC, and is called whenever the parent process sends a message via mad_decoder_message().

9.3.1 Detailed Description

Callback functions definitions for use in mad_decoder struct.

See also:

decoder::run_sync for details.

Definition in file callbacks.h.

9.3.2 Variable Documentation

9.3.2.1 enum mad_flow(∗ error_callback)(void ∗user_data, struct mad_stream ∗stream, struct mad_frame ∗frame)

The error callback function is called whenever a decoding error occurs.

The error is indicated by stream->error; the list of possible MAD_ERROR_∗ errors can be found in the stream.h header file.

Parameters:

user_data This can hold whatever you want – for example, song title, length, number of frames

stream the mad input stream the error occured in.

frame the mad frame the error occured in.

See also:

minimad::error() function for an example
decoder::error_default()

Referenced by run_sync().

9.3.2.2 enum mad_flow(∗ filter_callback)(void ∗user_data, struct mad_stream const ∗stream, struct mad_frame ∗)

The filter callback function is called after decoding a frame, but before synthesis.

Here it is possible to modify the frame's subband samples, for example to perform a uniform attenuation/amplification, or to do other special processing in the frequency domain.

Parameters:

user_data This can hold whatever you want

stream the mad input stream currently processed.

frame the MPEG frame just decoded.

9.3.2.3 enum mad_flow(∗ header_callback)(void ∗user_data, struct mad_header const ∗header)

The function called after MPEG headers have been decoded .

Parameters:

user_data This can hold whatever you want – for example, song title, length, number of frames

header the mad header just read.

9.3.2.4 enum mad_flow(∗ input_callback)(void ∗user_data, struct mad_stream ∗stream)

The purpose of this callback is to (re)fill the stream buffer which is to be decoded.

Parameters:

user_data This can hold whatever you want

stream the mad input stream you need to fill.

See also:

minimad::input() function for an example

9.3.2.5 enum mad_flow(∗ message_callback)(void ∗user_data, void ∗message, unsigned int ∗size)

The message callback function is only used with MAD_DECODER_MODE_ASYNC, and is called whenever the parent process sends a message via mad_decoder_message().

This callback can generate a reply by overwriting the message buffer that is passed to it. (The size of the reply must be the same or smaller than the message.)

Parameters:

user_data This can hold whatever you want

message the message to send

size of the message

See also:

decoder::check_message() , decoder::send()

9.3.2.6 enum mad_flow(∗ output_callback)(void ∗user_data, struct mad_header const ∗header, struct mad_pcm ∗pcm)

The purpose of this callback is to output (or play) the decoded PCM audio.

It is called after each frame of MPEG audio data has been completely decoded.

Parameters:

user_data This can hold whatever you want

header the stream header

pcm the audio data to output.

See also:

minimad::output() function for an example

9.4 config.dox File Reference

9.5 config.h File Reference

Generated from config.h.in by configure.

Defines

- #define HAVE_ASSERT_H 1
- #define HAVE_DLFCN_H 1
- #define HAVE_ERRNO_H 1
- #define HAVE_FCNTL 1
- #define HAVE_FCNTL_H 1
- #define HAVE_FORK 1
- #define HAVE_INTTYPES_H 1
- #define HAVE_LIMITS_H 1
- #define HAVE_MEMORY_H 1
- #define HAVE_PIPE 1
- #define HAVE_STDINT_H 1
- #define HAVE_STDLIB_H 1
- #define HAVE_STRINGS_H 1
- #define HAVE_STRING_H 1
- #define HAVE_SYS_STAT_H 1
- #define HAVE_SYS_TYPES_H 1
- #define HAVE_SYS_WAIT_H 1
- #define HAVE_UNISTD_H 1
- #define HAVE_WAITPID 1
- #define PACKAGE "libmad"
- #define PACKAGE_BUGREPORT "support at underbit com"
- #define PACKAGE_NAME "MPEG Audio Decoder"
- #define PACKAGE_STRING "MPEG Audio Decoder 0.15.1b"
- #define PACKAGE_TARNAME "libmad"
- #define PACKAGE_VERSION "0.15.1b"
- #define SIZEOF_INT 4
- #define SIZEOF_LONG 4
- #define SIZEOF_LONG_LONG 8
- #define STDC_HEADERS 1
- #define VERSION "0.15.1b"

9.5.1 Detailed Description

Generated from config.h.in by configure.

Definition in file config.h.

9.5.2 Define Documentation

9.5.2.1 #define HAVE_ASSERT_H 1

Definition at line 11 of file config.h.

9.5.2.2 #define HAVE_DLFCN_H 1

Definition at line 14 of file config.h.

9.5.2.3 #define HAVE_ERRNO_H 1

Definition at line 17 of file config.h.

9.5.2.4 #define HAVE_FCNTL 1

Definition at line 20 of file config.h.

9.5.2.5 #define HAVE_FCNTL_H 1

Definition at line 23 of file config.h.

9.5.2.6 #define HAVE_FORK 1

Definition at line 26 of file config.h.

9.5.2.7 #define HAVE_INTTYPES_H 1

Definition at line 29 of file config.h.

9.5.2.8 #define HAVE_LIMITS_H 1

Definition at line 32 of file config.h.

9.5.2.9 #define HAVE_MEMORY_H 1

Definition at line 41 of file config.h.

9.5.2.10 #define HAVE_PIPE 1

Definition at line 44 of file config.h.

9.5.2.11 #define HAVE_STDINT_H 1

Definition at line 47 of file config.h.

9.5.2.12 #define HAVE_STDLIB_H 1

Definition at line 50 of file config.h.

9.5.2.13 #define HAVE_STRING_H 1

Definition at line 56 of file config.h.

9.5.2.14 #define HAVE_STRINGS_H 1

Definition at line 53 of file config.h.

9.5.2.15 #define HAVE_SYS_STAT_H 1

Definition at line 59 of file config.h.

9.5.2.16 #define HAVE_SYS_TYPES_H 1

Definition at line 62 of file config.h.

9.5.2.17 #define HAVE_SYS_WAIT_H 1

Definition at line 65 of file config.h.

9.5.2.18 #define HAVE_UNISTD_H 1

Definition at line 68 of file config.h.

9.5.2.19 #define HAVE_WAITPID 1

Definition at line 71 of file config.h.

9.5.2.20 #define PACKAGE "libmad"

Definition at line 90 of file config.h.

9.5.2.21 #define PACKAGE_BUGREPORT "support at underbit com"

Definition at line 93 of file config.h.

9.5.2.22 #define PACKAGE_NAME "MPEG Audio Decoder"

Definition at line 96 of file config.h.

9.5.2.23 #define PACKAGE_STRING "MPEG Audio Decoder 0.15.1b"

Definition at line 99 of file config.h.

9.5.2.24 #define PACKAGE_TARNAME "libmad"

Definition at line 102 of file config.h.

9.5.2.25 #define PACKAGE_VERSION "0.15.1b"

Definition at line 105 of file config.h.

9.5.2.26 #define SIZEOF_INT 4

Definition at line 108 of file config.h.

9.5.2.27 #define SIZEOF_LONG 4

Definition at line 111 of file config.h.

9.5.2.28 #define SIZEOF_LONG_LONG 8

Definition at line 114 of file config.h.

9.5.2.29 #define STDC_HEADERS 1

Definition at line 117 of file config.h.

9.5.2.30 #define VERSION "0.15.1b"

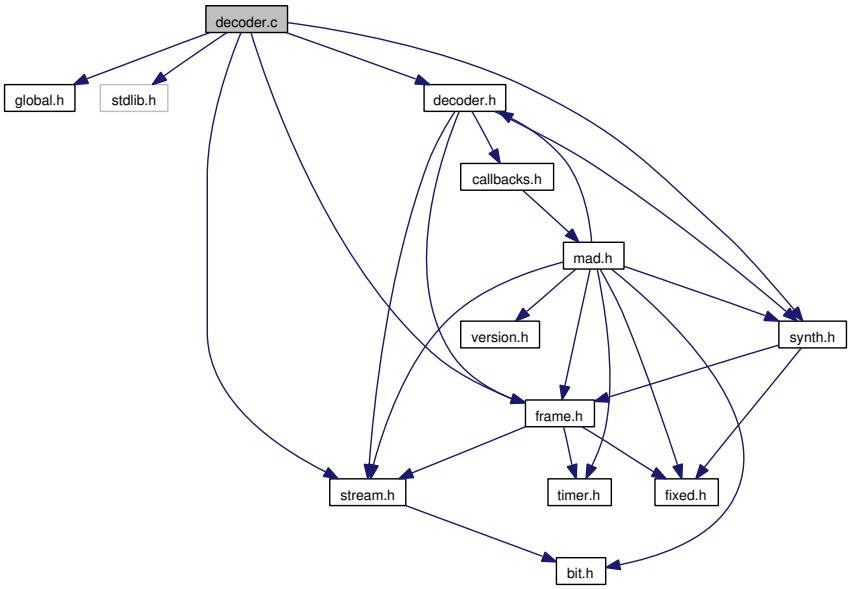
Definition at line 120 of file config.h.

9.6 decoder.c File Reference

Definition of the mad_decoder engine, the core of libmad library.

```
#include "global.h"
#include <stdlib.h>
#include "stream.h"
#include "frame.h"
#include "synth.h"
#include "decoder.h"
```

Include dependency graph for decoder.c:



Functions

- void mad_decoder_init (struct mad_decoder *decoder, void *data, input_callback input_func, header_callback header_func, filter_callback filter_func, output_callback output_func, error_callback error_func, message_callback message_func)

  initialize a decoder object with callback routines.

- int mad_decoder_finish (struct mad_decoder *decoder)
- static enum mad_flow error_default (void *data, struct mad_stream *stream, struct mad_frame *frame)
- static int run_sync (struct mad_decoder *decoder)
- int mad_decoder_run (struct mad_decoder *decoder, enum mad_decoder_mode mode)

run the decoder thread either synchronously or asynchronously.

- int mad_decoder_message (struct mad_decoder *decoder, void *message, unsigned int *len)

  send a message to and receive a reply from the decoder process.

9.6.1 Detailed Description

Definition of the mad_decoder engine, the core of libmad library.

Definition in file decoder.c.

9.6.2 Function Documentation

9.6.2.1 static enum mad_flow error_default (void * *data*, struct mad_stream * *stream*, struct mad_frame * *frame*) [static]

Definition at line 290 of file decoder.c.

References mad_stream::error, MAD_ERROR_BADCRC, MAD_FLOW_CONTINUE, MAD_FLOW_IGNORE, and mad_frame_mute().

Referenced by run_sync().

Here is the call graph for this function:

9.6.2.2 int mad_decoder_finish (struct mad_decoder * *decoder*)

Definition at line 90 of file decoder.c.

References mad_decoder::async, mad_decoder::in, MAD_DECODER_MODE_ASYNC, mad_decoder::mode, mad_decoder::out, and mad_decoder::pid.

Referenced by decode().

9.6.2.3 void mad_decoder_init (struct mad_decoder * *decoder*, void * *data*, input_callback *input_func*, header_callback *header_func*, filter_callback *filter_func*, output_callback *output_func*, error_callback *error_func*, message_callback *message_func*)

initialize a decoder object with callback routines.

Definition at line 62 of file decoder.c.

References mad_decoder::async, mad_decoder::cb_data, mad_decoder::error_-func, mad_decoder::filter_func, mad_decoder::header_func, mad_decoder::in, mad_decoder::input_func, mad_decoder::message_func, mad_decoder::mode, mad_-decoder::options, mad_decoder::out, mad_decoder::output_func, mad_decoder::pid, and mad_decoder::sync.

Referenced by decode().

9.6.2.4 int mad_decoder_message (struct mad_decoder * *decoder*, void * *message*, unsigned int * *len*)

send a message to and receive a reply from the decoder process.

Definition at line 561 of file decoder.c.

References mad_decoder::async, mad_decoder::in, MAD_DECODER_MODE_-ASYNC, MAD_FLOW_CONTINUE, mad_decoder::mode, and mad_decoder::out.

9.6.2.5 int mad_decoder_run (struct mad_decoder * *decoder*, enum mad_decoder_mode *mode*)

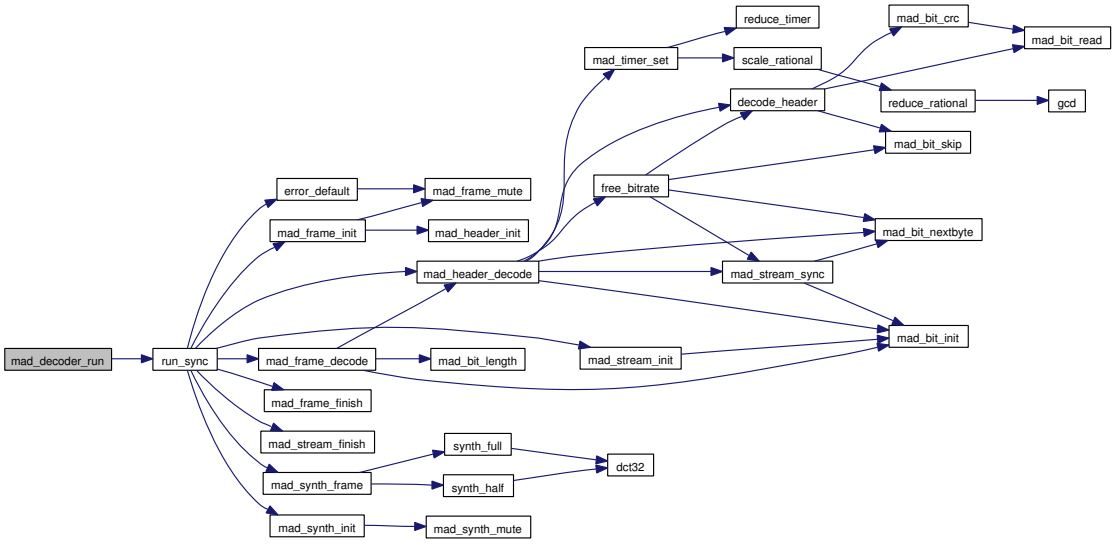
run the decoder thread either synchronously or asynchronously.

Definition at line 526 of file decoder.c.

References MAD_DECODER_MODE_ASYNC, MAD_DECODER_MODE_SYNC, mad_decoder::mode, run_sync(), and mad_decoder::sync.

Referenced by decode().

Here is the call graph for this function:



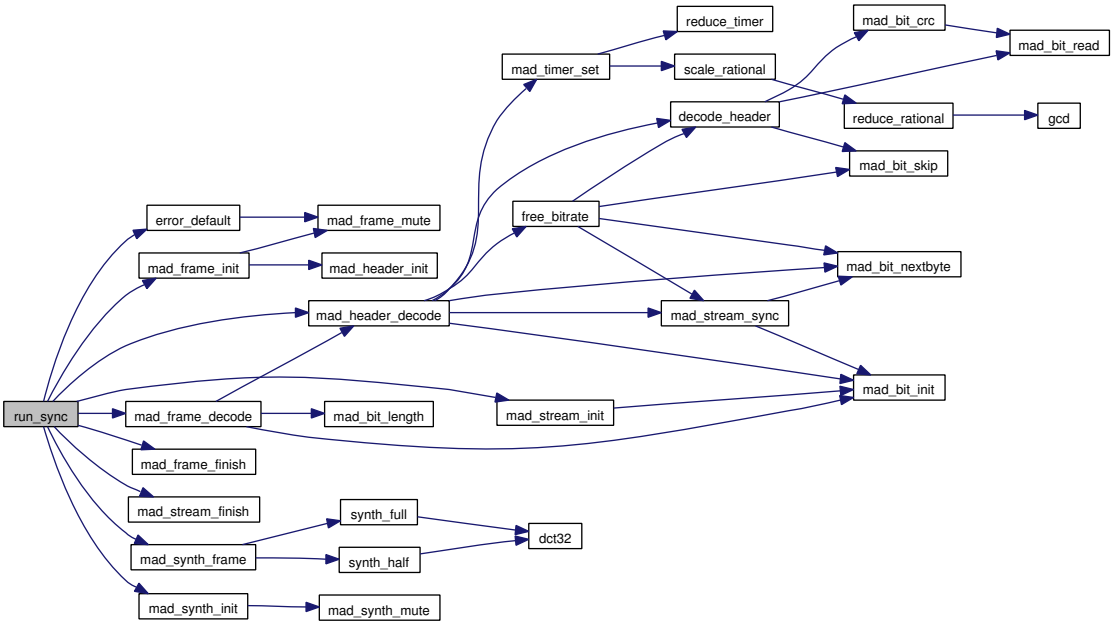
9.6.2.6 static int run_sync (struct mad_decoder * *decoder*) [static]

Definition at line 310 of file decoder.c.

References mad_decoder::cb_data, mad_stream::error, error_callback, error_default(), mad_decoder::error_func, mad_decoder::filter_func, mad_decoder::frame, mad_frame::header, mad_decoder::header_func, mad_decoder::input_func, MAD_DECODER_MODE_ASYNC, MAD_ERROR_BUFLEN, MAD_FLOW_BREAK, MAD_FLOW_CONTINUE, MAD_FLOW_IGNORE, MAD_FLOW_STOP, mad_frame_decode(), mad_frame_finish(), mad_frame_init(), mad_header_decode(), MAD_RECOVERABLE, mad_stream_finish(), mad_stream_init(), mad_stream_options, mad_synth_finish, mad_synth_frame(), mad_synth_init(), mad_decoder::mode, mad_decoder::options, mad_decoder::output_func, mad_synth::pcm, mad_decoder::stream, mad_decoder::sync, and mad_decoder::synth.

Referenced by mad_decoder_run().

Here is the call graph for this function:

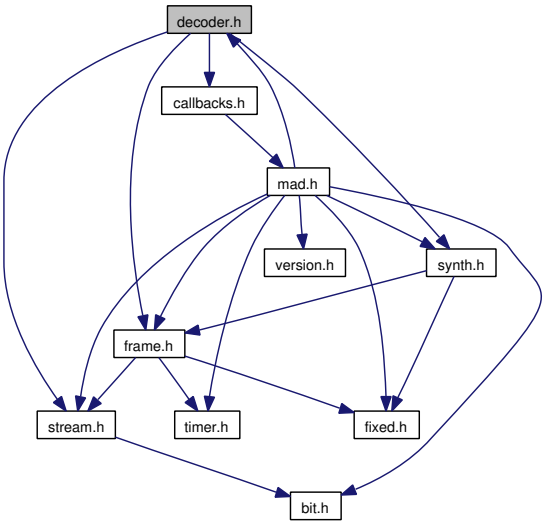


9.7 decoder.h File Reference

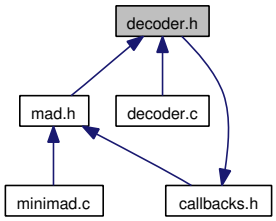
Declaration of the mad_decoder structure and methods, the core of libmad library.

```
#include "stream.h"
#include "frame.h"
#include "synth.h"
#include "callbacks.h"
```

Include dependency graph for decoder.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct mad_decoder

  This holds all information about how you want your stream decoded, such as input/output functions, error handling functions, etc.

Defines

- #define mad_decoder_options(decoder, opts) ((void) ((decoder) → options = (opts)))

Enumerations

- enum mad_decoder_mode { MAD_DECODER_MODE_SYNC = 0, MAD_DECODER_MODE_ASYNC }
- enum mad_flow { MAD_FLOW_CONTINUE = 0x0000, MAD_FLOW_STOP = 0x0010, MAD_FLOW_BREAK = 0x0011, MAD_FLOW_IGNORE = 0x0020 }

Functions

- void mad_decoder_init (struct mad_decoder *decoder, void *data, input_callback input_func, header_callback header_func, filter_callback filter_func, output_callback output_func, error_callback error_func, message_callback message_func)

  initialize a decoder object with callback routines.

- int mad_decoder_finish (struct mad_decoder *)
- int mad_decoder_run (struct mad_decoder *, enum mad_decoder_mode)

  run the decoder thread either synchronously or asynchronously.

- int mad_decoder_message (struct mad_decoder *, void *, unsigned int *)

  send a message to and receive a reply from the decoder process.

9.7.1 Detailed Description

Declaration of the mad_decoder structure and methods, the core of libmad library.

Definition in file decoder.h.

9.7.2 Define Documentation

9.7.2.1 #define mad_decoder_options(decoder, opts) ((void) ((decoder) → options = (opts)))

Definition at line 87 of file decoder.h.

9.7.3 Enumeration Type Documentation

9.7.3.1 enum mad_decoder_mode

Enumerator:

MAD_DECODER_MODE_SYNC

MAD_DECODER_MODE_ASYNC

Definition at line 33 of file decoder.h.

9.7.3.2 enum mad_flow

Enumerator:

MAD_FLOW_CONTINUE continue normally

MAD_FLOW_STOP stop decoding normally

MAD_FLOW_BREAK stop decoding and signal an error

MAD_FLOW_IGNORE ignore the current frame

Definition at line 38 of file decoder.h.

9.7.4 Function Documentation

9.7.4.1 int mad_decoder_finish (struct mad_decoder *)

Definition at line 90 of file decoder.c.

References mad_decoder::async, mad_decoder::in, MAD_DECODER_MODE_-ASYNC, mad_decoder::mode, mad_decoder::out, and mad_decoder::pid.

Referenced by decode().

9.7.4.2 void mad_decoder_init (struct mad_decoder * *decoder*, void * *data*, input_callback *input_func*, header_callback *header_func*, filter_callback *filter_func*, output_callback *output_func*, error_callback *error_func*, message_callback *message_func*)

initialize a decoder object with callback routines.

Definition at line 62 of file decoder.c.

References mad_decoder::async, mad_decoder::cb_data, mad_decoder::error_-func, mad_decoder::filter_func, mad_decoder::header_func, mad_decoder::in, mad_decoder::input_func, mad_decoder::message_func, mad_decoder::mode, mad_-decoder::options, mad_decoder::out, mad_decoder::output_func, mad_decoder::pid, and mad_decoder::sync.

Referenced by decode().

9.7.4.3 int mad_decoder_message (struct mad_decoder *, void *, unsigned int *)

send a message to and receive a reply from the decoder process.

Definition at line 561 of file decoder.c.

References mad_decoder::async, mad_decoder::in, MAD_DECODER_MODE_ASYNC, MAD_FLOW_CONTINUE, mad_decoder::mode, and mad_decoder::out.

9.7.4.4 int mad_decoder_run (struct mad_decoder *, enum *mad_decoder_mode*)

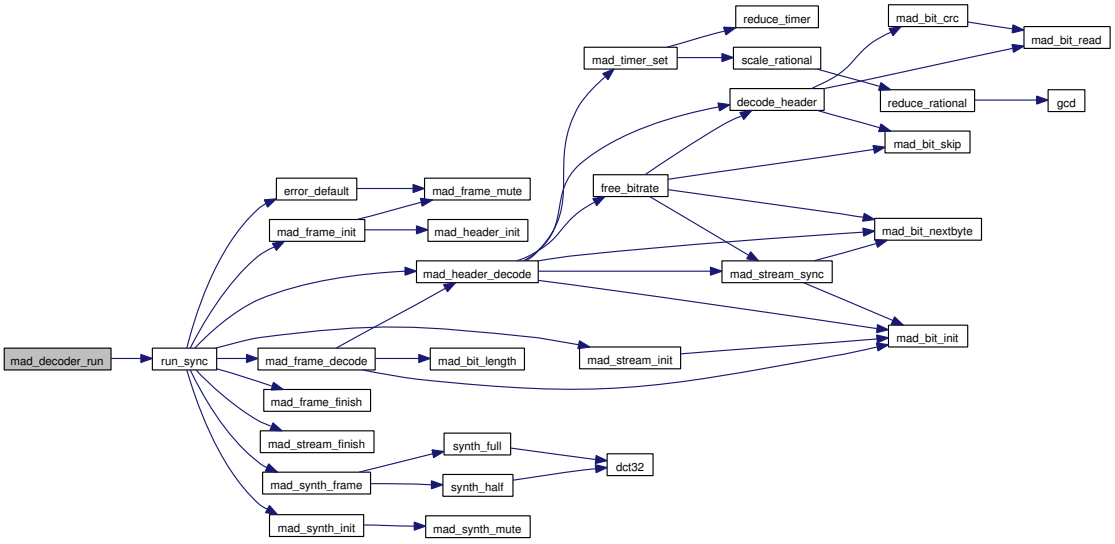
run the decoder thread either synchronously or asynchronously.

Definition at line 526 of file decoder.c.

References MAD_DECODER_MODE_ASYNC, MAD_DECODER_MODE_SYNC, mad_decoder::mode, run_sync(), and mad_decoder::sync.

Referenced by decode().

Here is the call graph for this function:

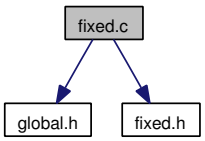


9.8 fixed.c File Reference

Implementation of some mad_fixed_t type operators.

```
#include "global.h"
#include "fixed.h"
```

Include dependency graph for fixed.c:



Functions

- mad_fixed_t mad_f_abs (mad_fixed_t x)
  return absolute value of a fixed-point number.
- mad_fixed_t mad_f_div (mad_fixed_t x, mad_fixed_t y)
  perform division using fixed-point math.

9.8.1 Detailed Description

Implementation of some mad_fixed_t type operators.

Definition in file fixed.c.

9.8.2 Function Documentation

9.8.2.1 mad_fixed_t mad_f_abs (mad_fixed_t *x*)

return absolute value of a fixed-point number.

Definition at line 36 of file fixed.c.

Referenced by mad_f_div().

9.8.2.2 mad_fixed_t mad_f_div (mad_fixed_t *x*, mad_fixed_t *y*)

perform division using fixed-point math.

Definition at line 44 of file fixed.c.

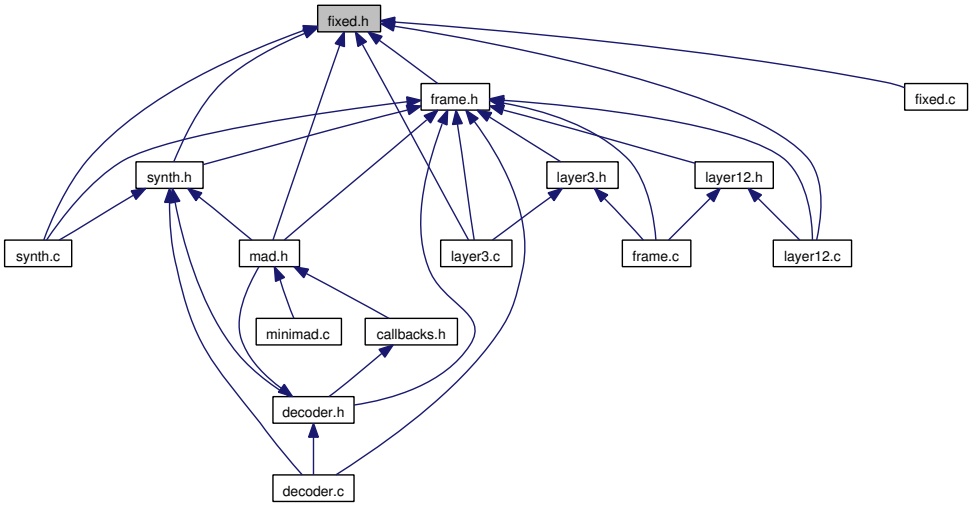
References mad_f_abs(), MAD_F_FRACBITS, mad_f_intpart, MAD_F_MAX, and MAD_F_MIN.

Here is the call graph for this function:

9.9 fixed.h File Reference

Fixed-point format definitions and tools.

This graph shows which files directly or indirectly include this file:



Defines

- #define mad_fixed64_t signed long long
- #define MAD_F_FRACBITS 28

  Changing the definition of MAD_F_FRACBITS is only partially supported, and must be done with care.

- #define MAD_F(x) ((mad_fixed_t) (x##L))
- #define MAD_F_MIN ((mad_fixed_t) -0x80000000L)
- #define MAD_F_MAX ((mad_fixed_t) +0x7fffffffL)
- #define MAD_F_ONE MAD_F(0x10000000)
- #define mad_f_tofixed(x)
- #define mad_f_todouble(x)
- #define mad_f_intpart(x) ((x) >> MAD_F_FRACBITS)
- #define mad_f_fracpart(x) ((x) & ((1L << MAD_F_FRACBITS) - 1))
- #define mad_f_fromint(x) ((x) << MAD_F_FRACBITS)
- #define mad_f_add(x, y) ((x) + (y))
- #define mad_f_sub(x, y) ((x) - (y))
- #define mad_f_mul(x, y)
- #define MAD_F_ML0(hi, lo, x, y) ((lo) = mad_f_mul((x), (y)))
- #define MAD_F_MLA(hi, lo, x, y) ((lo) += mad_f_mul((x), (y)))
- #define MAD_F_MLN(hi, lo) ((lo) = -(lo))
- #define MAD_F_MLZ(hi, lo) ((void) (hi), (mad_fixed_t) (lo))
- #define mad_f_scale64(hi, lo)
- #define MAD_F_SCALEBITS MAD_F_FRACBITS

Typedefs

- typedef signed long mad_fixed_t
- typedef signed long mad_fixed64hi_t
- typedef unsigned long mad_fixed64lo_t
- typedef mad_fixed_t mad_sample_t

Functions

- mad_fixed_t mad_f_abs (mad_fixed_t)

  return absolute value of a fixed-point number.

- mad_fixed_t mad_f_div (mad_fixed_t, mad_fixed_t)

  perform division using fixed-point math.

9.9.1 Detailed Description

Fixed-point format definitions and tools.

Fixed-point format: 0xABBBBBBB

```
A == whole part      (sign + 3 bits)
B == fractional part (28 bits)
```

Values are signed two's complement, so the effective range is:

```
0x80000000 to 0x7fffffff
      -8.0 to +7.999999996274709701538085937 5
```

The smallest representable value is:

```
0x00000001 == 0.000000003725290298461914062 5 (i.e. about 3.725e-9)
```

28 bits of fractional accuracy represent about 8.6 digits of decimal accuracy.

Fixed-point numbers can be added or subtracted as normal integers, but multiplication requires shifting the 64-bit result from 56 fractional bits back to 28 (and rounding.)

Definition in file fixed.h.

9.9.2 Define Documentation

9.9.2.1 #define MAD_F(x) ((mad_fixed_t) (x##L))

Definition at line 85 of file fixed.h.

Referenced by dctIV(), fastsdct(), and sdctII().

9.9.2.2 #define mad_f_add(x, y) ((x) + (y))

Definition at line 116 of file fixed.h.

9.9.2.3 #define MAD_F_FRACBITS 28

Changing the definition of MAD_F_FRACBITS is only partially supported, and must be done with care.

Definition at line 82 of file fixed.h.

Referenced by I_sample(), II_samples(), mad_f_div(), and scale().

9.9.2.4 #define mad_f_fracpart(x) ((x) & ((1L << MAD_F_FRACBITS) - 1))

Definition at line 111 of file fixed.h.

9.9.2.5 #define mad_f_fromint(x) ((x) << MAD_F_FRACBITS)

Definition at line 114 of file fixed.h.

9.9.2.6 #define mad_f_intpart(x) ((x) >> MAD_F_FRACBITS)

Definition at line 110 of file fixed.h.

Referenced by mad_f_div().

9.9.2.7 #define MAD_F_MAX ((mad_fixed_t) +0x7fffffffL)

Definition at line 101 of file fixed.h.

Referenced by III_requantize(), and mad_f_div().

9.9.2.8 #define MAD_F_MIN ((mad_fixed_t) -0x80000000L)

Definition at line 100 of file fixed.h.

Referenced by mad_f_div().

9.9.2.9 #define MAD_F_ML0(hi, lo, x, y) ((lo) = mad_f_mul((x), (y)))

Definition at line 471 of file fixed.h.

Referenced by III_aliasreduce(), and III_imdct_s().

9.9.2.10 #define MAD_F_MLA(hi, lo, x, y) ((lo) += mad_f_mul((x), (y)))

Definition at line 472 of file fixed.h.

Referenced by III_aliasreduce(), and III_imdct_s().

9.9.2.11 #define MAD_F_MLN(hi, lo) ((lo) = -(lo))

Definition at line 473 of file fixed.h.

9.9.2.12 #define MAD_F_MLZ(hi, lo) ((void) (hi), (mad_fixed_t) (lo))

Definition at line 474 of file fixed.h.

Referenced by III_aliasreduce(), and III_imdct_s().

9.9.2.13 #define mad_f_mul(x, y)

Value:

```
({ register mad_fixed64hi_t __hi;  \
       register mad_fixed64lo_t __lo;  \
       MAD_F_MLX(__hi, __lo, (x), (y));  \
       mad_f_scale64(__hi, __lo);  \
    })
```

Definition at line 462 of file fixed.h.

Referenced by dctIV(), fastsdct(), I_sample(), II_samples(), III_imdct_l(), III_imdct_s(), III_requantize(), III_stereo(), mad_layer_I(), mad_layer_II(), and sdctII().

9.9.2.14 #define MAD_F_ONE MAD_F(0x10000000)

Definition at line 103 of file fixed.h.

Referenced by I_sample(), and scale().

9.9.2.15 #define mad_f_scale64(hi, lo)

Value:

```
((mad_fixed_t)  \
     (((hi) << (32 - MAD_F_SCALEBITS)) |  \
      ((lo) >> MAD_F_SCALEBITS)))
```

Definition at line 496 of file fixed.h.

9.9.2.16 #define MAD_F_SCALEBITS MAD_F_FRACBITS

Definition at line 501 of file fixed.h.

9.9.2.17 #define mad_f_sub(x, y) ((x) - (y))

Definition at line 117 of file fixed.h.

9.9.2.18 #define mad_f_todouble(x)

Value:

```
((double)  \
                                    ((x) / (double) (1L << MAD_F_FRACBITS)))
```

Definition at line 107 of file fixed.h.

Referenced by III_requantize().

9.9.2.19 #define mad_f_tofixed(x)

Value:

```
((mad_fixed_t)  \
                                    ((x) * (double) (1L << MAD_F_FRACBITS) + 0.5))
```

Definition at line 105 of file fixed.h.

9.9.2.20 #define mad_fixed64_t signed long long

Definition at line 40 of file fixed.h.

9.9.3 Typedef Documentation

9.9.3.1 typedef signed long mad_fixed64hi_t

Definition at line 33 of file fixed.h.

9.9.3.2 typedef unsigned long mad_fixed64lo_t

Definition at line 34 of file fixed.h.

9.9.3.3 typedef signed long mad_fixed_t

Definition at line 31 of file fixed.h.

9.9.3.4 typedef mad_fixed_t mad_sample_t

Definition at line 46 of file fixed.h.

9.9.4 Function Documentation

9.9.4.1 mad_fixed_t mad_f_abs (mad_fixed_t)

return absolute value of a fixed-point number.

Definition at line 36 of file fixed.c.

Referenced by mad_f_div().

9.9.4.2 mad_fixed_t mad_f_div (mad_fixed_t, mad_fixed_t)

perform division using fixed-point math.

Definition at line 44 of file fixed.c.

References mad_f_abs(), MAD_F_FRACBITS, mad_f_intpart, MAD_F_MAX, and MAD_F_MIN.

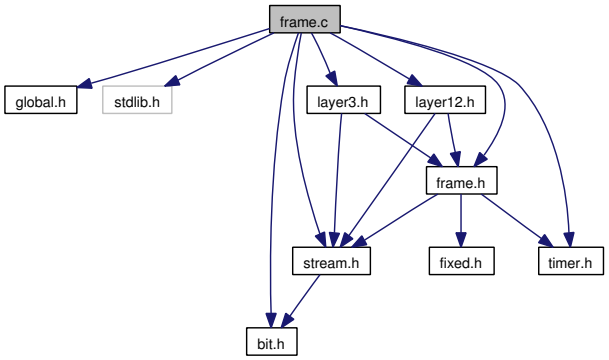
Here is the call graph for this function:

9.10 frame.c File Reference

Implementation of the mad_frame and mad_header methods.

```
#include "global.h"
#include <stdlib.h>
#include "bit.h"
#include "stream.h"
#include "frame.h"
#include "timer.h"
#include "layer12.h"
#include "layer3.h"
```

Include dependency graph for frame.c:



Functions

- void mad_header_init (struct mad_header *header)

  Initializes a newly created mad_header structure.

- void mad_frame_init (struct mad_frame *frame)

  Initializes a frame struct.

- void mad_frame_finish (struct mad_frame *frame)

  Deallocates any dynamic memory associated with frame.

- static int decode_header (struct mad_header *header, struct mad_stream *stream)

  Reads header data and following CRC word.

- static int free_bitrate (struct mad_stream *stream, struct mad_header const *header)

Attempts to discover the bitstream's free bitrate.

- int mad_header_decode (struct mad_header *header, struct mad_stream *stream)

  Reads the next frame header from the stream.

- int mad_frame_decode (struct mad_frame *frame, struct mad_stream *stream)

  Decodes a single frame from a bitstream.

- void mad_frame_mute (struct mad_frame *frame)

  Zeroes all subband values so the frame becomes silent.

Variables

- static unsigned long const bitrate_table [5][15]
- static unsigned int const samplerate_table [3] = { 44100, 48000, 32000 }
- static int(*const decoder_table [3])(struct mad_stream *, struct mad_frame *)

9.10.1 Detailed Description

Implementation of the mad_frame and mad_header methods.

Definition in file frame.c.

9.10.2 Function Documentation

9.10.2.1 static int decode_header (struct mad_header * *header*, struct mad_stream * *stream*) [static]

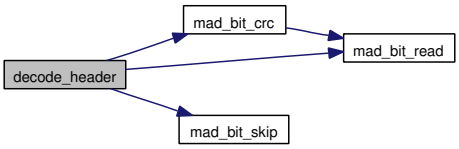
Reads header data and following CRC word.

Definition at line 119 of file frame.c.

References mad_header::bitrate, bitrate_table, mad_header::crc_check, mad_header::crc_target, mad_header::emphasis, mad_stream::error, mad_header::flags, mad_header::layer, mad_bit_crc(), mad_bit_read(), mad_bit_skip(), MAD_EMPHASIS_RESERVED, MAD_ERROR_BADBITRATE, MAD_ERROR_BADEMPHASIS, MAD_ERROR_BADLAYER, MAD_ERROR_BADSAMPLERATE, MAD_ERROR_LOSTSYNC, MAD_FLAG_COPYRIGHT, MAD_FLAG_LSF_EXT, MAD_FLAG_MPEG_2_5_EXT, MAD_FLAG_ORIGINAL, MAD_FLAG_PADDING, MAD_FLAG_PROTECTION, MAD_PRIVATE_HEADER, mad_header::mode, mad_header::mode_extension, mad_header::private_bits, mad_stream::ptr, mad_header::samplerate, and samplerate_table.

Referenced by free_bitrate(), and mad_header_decode().

Here is the call graph for this function:



9.10.2.2 static int free_bitrate (struct mad_stream * *stream*, struct mad_header const * *header*) [static]

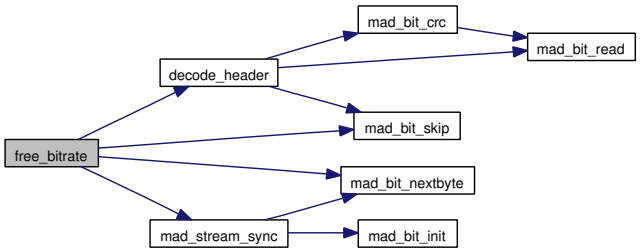
Attempts to discover the bitstream's free bitrate.

Definition at line 237 of file frame.c.

References decode_header(), mad_stream::error, mad_header::flags, mad_stream::freerate, mad_header::layer, mad_bit_nextbyte(), mad_bit_skip(), MAD_ERROR_LOSTSYNC, MAD_FLAG_LSF_EXT, MAD_FLAG_PADDING, MAD_LAYER_I, MAD_LAYER_III, mad_stream_sync(), mad_stream::ptr, mad_header::samplerate, and mad_stream::this_frame.

Referenced by mad_header_decode().

Here is the call graph for this function:



9.10.2.3 int mad_frame_decode (struct mad_frame * *frame*, struct mad_stream * *stream*)

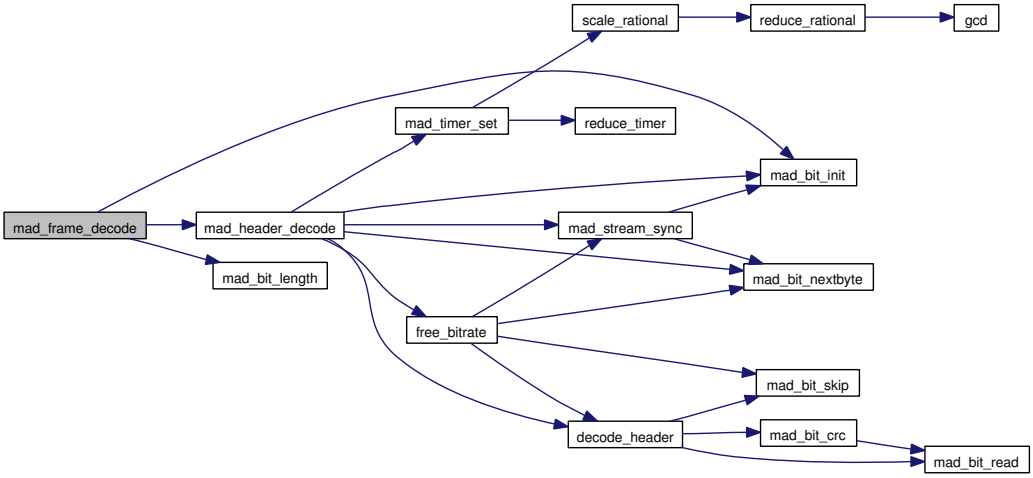
Decodes a single frame from a bitstream.

Definition at line 434 of file frame.c.

References mad_stream::anc_bitlen, mad_stream::anc_ptr, decoder_table, mad_stream::error, mad_header::flags, mad_frame::header, mad_header::layer, mad_bit_finish, mad_bit_init(), mad_bit_length(), MAD_FLAG_INCOMPLETE, mad_header_decode(), MAD_LAYER_III, MAD_RECOVERABLE, mad_stream::next_frame, mad_stream::options, mad_frame::options, mad_stream::ptr, and mad_stream::this_frame.

Referenced by run_sync().

Here is the call graph for this function:



9.10.2.4 void mad_frame_finish (struct mad_frame * *frame*)

Deallocates any dynamic memory associated with frame.

Definition at line 105 of file frame.c.

References mad_frame::header, mad_header_finish, and mad_frame::overlap.

Referenced by run_sync().

9.10.2.5 void mad_frame_init (struct mad_frame * *frame*)

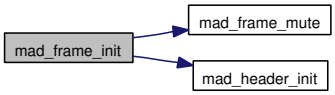
Initializes a frame struct.

Definition at line 92 of file frame.c.

References mad_frame::header, mad_frame_mute(), mad_header_init(), mad_frame::options, and mad_frame::overlap.

Referenced by run_sync().

Here is the call graph for this function:



9.10.2.6 void mad_frame_mute (struct mad_frame * *frame*)

Zeroes all subband values so the frame becomes silent.

Definition at line 479 of file frame.c.

References mad_frame::overlap, s, and mad_frame::sbsample.

Referenced by error_default(), and mad_frame_init().

9.10.2.7 int mad_header_decode (struct mad_header * *header*, struct mad_stream * *stream*)

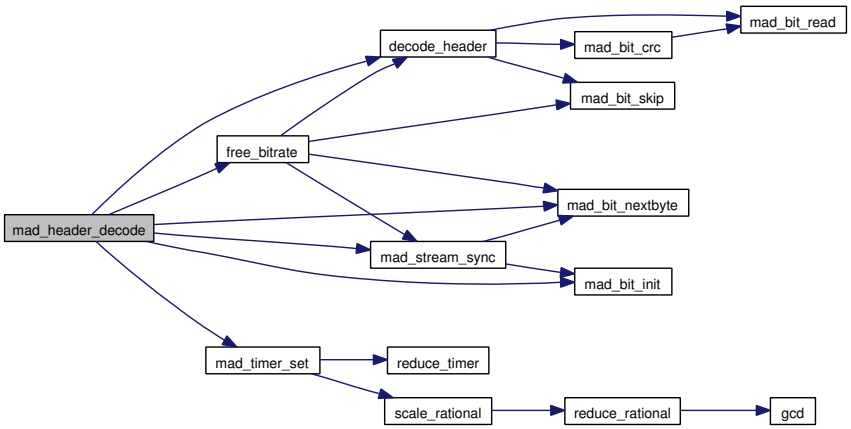
Reads the next frame header from the stream.

Definition at line 297 of file frame.c.

References mad_header::bitrate, mad_stream::bufend, decode_header(), mad_header::duration, mad_stream::error, mad_header::flags, free_bitrate(), mad_stream::freerate, mad_header::layer, mad_bit_init(), mad_bit_nextbyte(), MAD_BUFFER_GUARD, MAD_ERROR_BUFLEN, MAD_ERROR_BUFPTR, MAD_ERROR_LOSTSYNC, MAD_FLAG_FREEFORMAT, MAD_FLAG_INCOMPLETE, MAD_FLAG_LSF_EXT, MAD_FLAG_PADDING, MAD_LAYER_I, MAD_LAYER_III, MAD_NSBSAMPLES, mad_stream_sync(), mad_timer_set(), mad_stream::next_frame, mad_stream::ptr, mad_header::samplerate, mad_stream::skiplen, mad_stream::sync, and mad_stream::this_frame.

Referenced by mad_frame_decode(), and run_sync().

Here is the call graph for this function:



9.10.2.8 void mad_header_init (struct mad_header * *header*)

Initializes a newly created mad_header structure.

Definition at line 70 of file frame.c.

References mad_header::bitrate, mad_header::crc_check, mad_header::crc_target, mad_header::duration, mad_header::emphasis, mad_header::flags, mad_header::layer, mad_timer_zero, mad_header::mode, mad_header::mode_extension,

mad_header::private_bits, and mad_header::samplerate.

Referenced by mad_frame_init().

9.10.3 Variable Documentation

9.10.3.1 unsigned long const bitrate_table[5][15] [static]

Initial value:

```
 {

  { 0,  32000,  64000,  96000, 128000, 160000, 192000, 224000,
       256000, 288000, 320000, 352000, 384000, 416000, 448000 },
  { 0,  32000,  48000,  56000,  64000,  80000,  96000, 112000,
       128000, 160000, 192000, 224000, 256000, 320000, 384000 },
  { 0,  32000,  40000,  48000,  56000,  64000,  80000,  96000,
       112000, 128000, 160000, 192000, 224000, 256000, 320000 },


  { 0,  32000,  48000,  56000,  64000,  80000,  96000, 112000,
       128000, 144000, 160000, 176000, 192000, 224000, 256000 },
  { 0,   8000,  16000,  24000,  32000,  40000,  48000,  56000,
        64000,  80000,  96000, 112000, 128000, 144000, 160000 }
}
```

Definition at line 41 of file frame.c.

Referenced by decode_header().

9.10.3.2 int(*const decoder_table[3])(struct mad_stream *, struct mad_frame *) [static]

Initial value:

```
 {
  mad_layer_I,
  mad_layer_II,
  mad_layer_III
}
```

Referenced by mad_frame_decode().

9.10.3.3 unsigned int const samplerate_table[3] = { 44100, 48000, 32000 } [static]

Definition at line 58 of file frame.c.

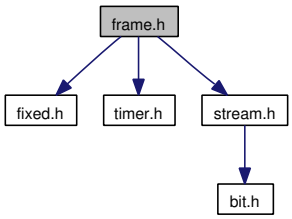
Referenced by decode_header().

9.11 frame.h File Reference

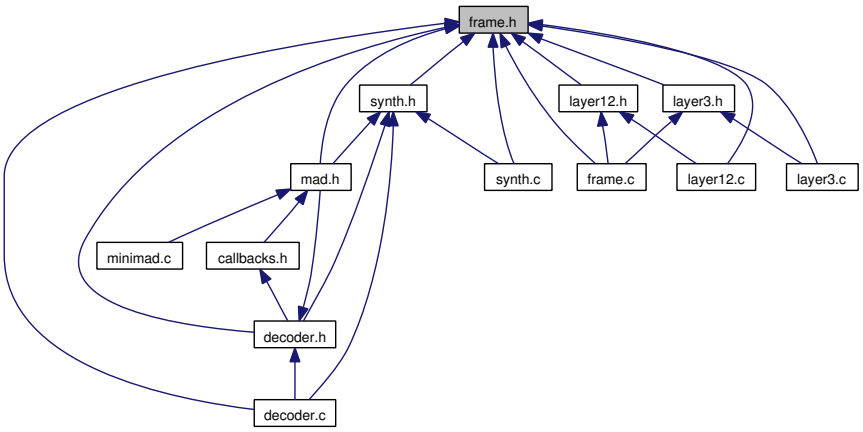
Declaration of the mad_frame and mad_header structures, as libmad decoding units, and associated methods.

```
#include "fixed.h"
#include "timer.h"
#include "stream.h"
```

Include dependency graph for frame.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct mad_header

  MPEG frame header information.

- struct mad_frame

  MPEG unit of encoding, and associated context.

Defines

- #define MAD_NCHANNELS(header) ((header) → mode ? 2 : 1)
- #define MAD_NSBSAMPLES(header)
- #define mad_header_finish(header)

  Releases a mad_header structure after usage.

Enumerations

- enum mad_layer { MAD_LAYER_I = 1, MAD_LAYER_II = 2, MAD_LAYER_III = 3 }
- enum mad_mode { MAD_MODE_SINGLE_CHANNEL = 0, MAD_MODE_DUAL_CHANNEL = 1, MAD_MODE_JOINT_STEREO = 2, MAD_MODE_STEREO = 3 }
- enum mad_emphasis { MAD_EMPHASIS_NONE = 0, MAD_EMPHASIS_50_15_US = 1, MAD_EMPHASIS_CCITT_J_17 = 3, MAD_EMPHASIS_RESERVED = 2 }
- enum {

  MAD_FLAG_NPRIVATE_III = 0x0007, MAD_FLAG_INCOMPLETE = 0x0008, MAD_FLAG_PROTECTION = 0x0010, MAD_FLAG_COPYRIGHT = 0x0020,

  MAD_FLAG_ORIGINAL = 0x0040, MAD_FLAG_PADDING = 0x0080, MAD_FLAG_I_STEREO = 0x0100, MAD_FLAG_MS_STEREO = 0x0200,

  MAD_FLAG_FREEFORMAT = 0x0400, MAD_FLAG_LSF_EXT = 0x1000, MAD_FLAG_MC_EXT = 0x2000, MAD_FLAG_MPEG_2_5_EXT = 0x4000 }
- enum { MAD_PRIVATE_HEADER = 0x0100, MAD_PRIVATE_III = 0x001f }

Functions

- void mad_header_init (struct mad_header *)

  Initializes a newly created mad_header structure.

- int mad_header_decode (struct mad_header *, struct mad_stream *)

  Reads the next frame header from the stream.

- void mad_frame_init (struct mad_frame *)

  Initializes a frame struct.

- void mad_frame_finish (struct mad_frame *)

  Deallocates any dynamic memory associated with frame.

- int mad_frame_decode (struct mad_frame *, struct mad_stream *)

  Decodes a single frame from a bitstream.

- void mad_frame_mute (struct mad_frame *)

  Zeroes all subband values so the frame becomes silent.

9.11.1 Detailed Description

Declaration of the mad_frame and mad_header structures, as libmad decoding units, and associated methods.

Definition in file frame.h.

9.11.2 Define Documentation

9.11.2.1 #define mad_header_finish(header)

Releases a mad_header structure after usage.

nothing

Definition at line 119 of file frame.h.

Referenced by mad_frame_finish().

9.11.2.2 #define MAD_NCHANNELS(header) ((header) → mode ? 2 : 1)

Definition at line 85 of file frame.h.

Referenced by mad_layer_I(), mad_layer_II(), mad_layer_III(), and mad_synth_frame().

9.11.2.3 #define MAD_NSBSAMPLES(header)

Value:

```
((header)->layer == MAD_LAYER_I ? 12 :  \
   (((header)->layer == MAD_LAYER_III &&  \
     ((header)->flags & MAD_FLAG_LSF_EXT)) ? 18 : 36))
```

Definition at line 86 of file frame.h.

Referenced by mad_header_decode(), and mad_synth_frame().

9.11.3 Enumeration Type Documentation

9.11.3.1 anonymous enum

Enumerator:

MAD_FLAG_NPRIVATE_III number of Layer III private bits

MAD_FLAG_INCOMPLETE header but not data is decoded
MAD_FLAG_PROTECTION frame has CRC protection
MAD_FLAG_COPYRIGHT frame is copyright
MAD_FLAG_ORIGINAL frame is original (else copy)
MAD_FLAG_PADDING frame has additional slot
MAD_FLAG_I_STEREO uses intensity joint stereo
MAD_FLAG_MS_STEREO uses middle/side joint stereo
MAD_FLAG_FREEFORMAT uses free format bitrate
MAD_FLAG_LSF_EXT lower sampling freq. extension
MAD_FLAG_MC_EXT multichannel audio extension
MAD_FLAG_MPEG_2_5_EXT MPEG 2.5 (unofficial) extension.

Definition at line 91 of file frame.h.

9.11.3.2 anonymous enum

Enumerator:

MAD_PRIVATE_HEADER header private bit
MAD_PRIVATE_III Layer III private bits (up to 5).

Definition at line 109 of file frame.h.

9.11.3.3 enum mad_emphasis

Enumerator:

MAD_EMPHASIS_NONE no emphasis
MAD_EMPHASIS_50_15_US 50/15 microseconds emphasis
MAD_EMPHASIS_CCITT_J_17 CCITT J.17 emphasis.
MAD_EMPHASIS_RESERVED unknown emphasis

Definition at line 45 of file frame.h.

9.11.3.4 enum mad_layer

Enumerator:

MAD_LAYER_I Layer I.
MAD_LAYER_II Layer II.
MAD_LAYER_III Layer III.

Definition at line 32 of file frame.h.

9.11.3.5 enum mad_mode

Enumerator:

MAD_MODE_SINGLE_CHANNEL single channel
MAD_MODE_DUAL_CHANNEL dual channel
MAD_MODE_JOINT_STEREO joint (MS/intensity) stereo
MAD_MODE_STEREO normal LR stereo

Definition at line 38 of file frame.h.

9.11.4 Function Documentation

9.11.4.1 int mad_frame_decode (struct mad_frame *, struct mad_stream *)

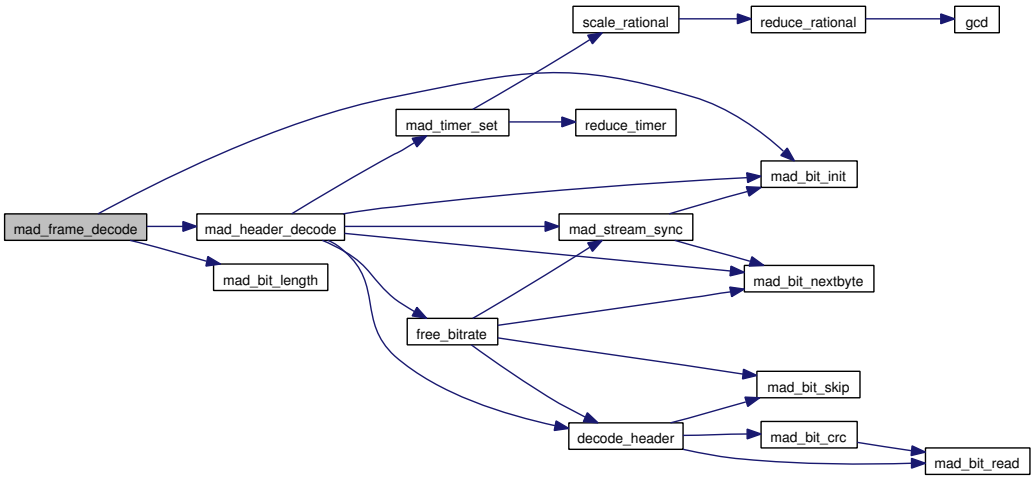
Decodes a single frame from a bitstream.

Definition at line 434 of file frame.c.

References mad_stream::anc_bitlen, mad_stream::anc_ptr, decoder_table, mad_stream::error, mad_header::flags, mad_frame::header, mad_header::layer, mad_bit_finish, mad_bit_init(), mad_bit_length(), MAD_FLAG_INCOMPLETE, mad_header_decode(), MAD_LAYER_III, MAD_RECOVERABLE, mad_stream::next_frame, mad_stream::options, mad_frame::options, mad_stream::ptr, and mad_stream::this_frame.

Referenced by run_sync().

Here is the call graph for this function:



9.11.4.2 void mad_frame_finish (struct mad_frame *)

Deallocates any dynamic memory associated with frame.

Definition at line 105 of file frame.c.

References mad_frame::header, mad_header_finish, and mad_frame::overlap.

Referenced by run_sync().

9.11.4.3 void mad_frame_init (struct mad_frame *)

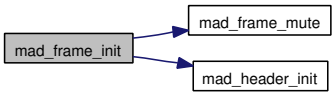
Initializes a frame struct.

Definition at line 92 of file frame.c.

References mad_frame::header, mad_frame_mute(), mad_header_init(), mad_frame::options, and mad_frame::overlap.

Referenced by run_sync().

Here is the call graph for this function:



9.11.4.4 void mad_frame_mute (struct mad_frame *)

Zeroes all subband values so the frame becomes silent.

Definition at line 479 of file frame.c.

References mad_frame::overlap, s, and mad_frame::sbsample.

Referenced by error_default(), and mad_frame_init().

9.11.4.5 int mad_header_decode (struct mad_header *, struct mad_stream *)

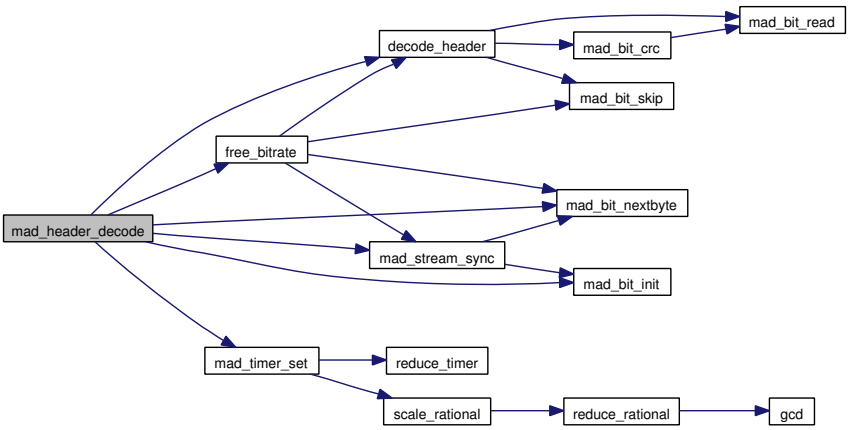
Reads the next frame header from the stream.

Definition at line 297 of file frame.c.

References mad_header::bitrate, mad_stream::bufend, decode_header(), mad_header::duration, mad_stream::error, mad_header::flags, free_bitrate(), mad_stream::freerate, mad_header::layer, mad_bit_init(), mad_bit_nextbyte(), MAD_BUFFER_GUARD, MAD_ERROR_BUFLEN, MAD_ERROR_BUFPTR, MAD_ERROR_LOSTSYNC, MAD_FLAG_FREEFORMAT, MAD_FLAG_INCOMPLETE, MAD_FLAG_LSF_EXT, MAD_FLAG_PADDING, MAD_LAYER_I, MAD_LAYER_III, MAD_NSBSAMPLES, mad_stream_sync(), mad_timer_set(), mad_stream::next_frame, mad_stream::ptr, mad_header::samplerate, mad_stream::skiplen, mad_stream::sync, and mad_stream::this_frame.

Referenced by mad_frame_decode(), and run_sync().

Here is the call graph for this function:



9.11.4.6 void mad_header_init (struct mad_header *)

Initializes a newly created mad_header structure.

Definition at line 70 of file frame.c.

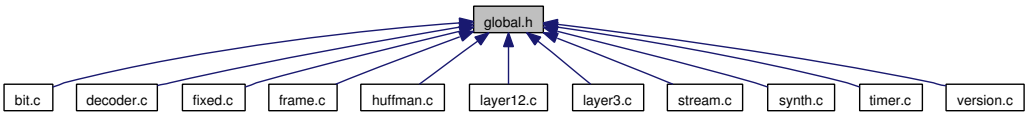
References mad_header::bitrate, mad_header::crc_check, mad_header::crc_target, mad_header::duration, mad_header::emphasis, mad_header::flags, mad_header::layer, mad_timer_zero, mad_header::mode, mad_header::mode_extension, mad_header::private_bits, and mad_header::samplerate.

Referenced by mad_frame_init().

9.12 global.h File Reference

Global defines.

This graph shows which files directly or indirectly include this file:



Defines

- #define assert(x) do { if (!(x)) abort(); } while (0)

9.12.1 Detailed Description

Global defines.

Checks configuration options validity and activates conditional features.

Definition in file global.h.

9.12.2 Define Documentation

9.12.2.1 #define assert(x) do { if (!(x)) abort(); } while (0)

Definition at line 58 of file global.h.

Referenced by III_huffdecode(), mad_layer_III(), reduce_rational(), and scale_-rational().

9.13 highlevel.dox File Reference

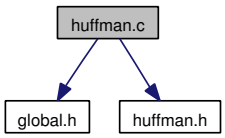
9.14 huffman.c File Reference

Declares some macros and initializes some data used in MPEG Layer III Huffman compression decoding.

```
#include "global.h"
```

```
#include "huffman.h"
```

Include dependency graph for huffman.c:



Defines

- #define PTR(offs, bits) { { 0, bits, offs } }
- #define V(v, w, x, y, hlen)
- #define PTR(offs, bits) { { 0, bits, offs } }
- #define V(x, y, hlen) { { 1, hlen, (x << 0) | (y << 4) } }

Variables

- static union huffquad const hufftabA [ ]
- static union huffquad const hufftabB [ ]
- static union huffpair const hufftab0 [ ]
- static union huffpair const hufftab1 [ ]
- static union huffpair const hufftab2 [ ]
- static union huffpair const hufftab3 [ ]
- static union huffpair const hufftab5 [ ]
- static union huffpair const hufftab6 [ ]
- static union huffpair const hufftab7 [ ]
- static union huffpair const hufftab8 [ ]
- static union huffpair const hufftab9 [ ]
- static union huffpair const hufftab10 [ ]
- static union huffpair const hufftab11 [ ]
- static union huffpair const hufftab12 [ ]
- static union huffpair const hufftab13 [ ]
- static union huffpair const hufftab15 [ ]
- static union huffpair const hufftab16 [ ]
- static union huffpair const hufftab24 [ ]
- union huffquad const *const mad_huff_quad_table [2] = { hufftabA, hufftabB }
- struct hufftable const mad_huff_pair_table [32]

9.14.1 Detailed Description

Declares some macros and initializes some data used in MPEG Layer III Huffman compression decoding.

Definition in file huffman.c.

9.14.2 Define Documentation

9.14.2.1 #define PTR(offs, bits) { { 0, bits, offs } }

Definition at line 124 of file huffman.c.

9.14.2.2 #define PTR(offs, bits) { { 0, bits, offs } }

Definition at line 124 of file huffman.c.

9.14.2.3 #define V(x, y, hlen) { { 1, hlen, (x << 0) | (y << 4) } }

Definition at line 128 of file huffman.c.

9.14.2.4 #define V(v, w, x, y, hlen)

Value:

```
{ { 1, hlen, (v <<  0) | (w <<  1) |  \
                                              (x <<  2) | (y <<  3) } }
```

Definition at line 128 of file huffman.c.

9.14.3 Variable Documentation

9.14.3.1 union huffpair const hufftab0[ ] [static]

Initial value:

```
 {
   V(0, 0, 0)
}
```

Definition at line 133 of file huffman.c.

9.14.3.2 union huffpair const hufftab1[ ] [static]

Initial value:

```
 {
   V(1, 1, 3),
   V(0, 1, 3),
   V(1, 0, 2),
   V(1, 0, 2),
   V(0, 0, 1),
   V(0, 0, 1),
   V(0, 0, 1),
   V(0, 0, 1)
}
```

Definition at line 138 of file huffman.c.

9.14.3.3 union huffpair const hufftab10[ ] [static]

Definition at line 588 of file huffman.c.

9.14.3.4 union huffpair const hufftab11[ ] [static]

Definition at line 726 of file huffman.c.

9.14.3.5 union huffpair const hufftab12[ ] [static]

Definition at line 862 of file huffman.c.

9.14.3.6 union huffpair const hufftab13[ ] [static]

Definition at line 990 of file huffman.c.

9.14.3.7 union huffpair const hufftab15[ ] [static]

Definition at line 1516 of file huffman.c.

9.14.3.8 union huffpair const hufftab16[ ] [static]

Definition at line 2024 of file huffman.c.

9.14.3.9 union huffpair const hufftab2[ ] [static]

Initial value:

```
 {
   PTR(8, 3),
   V(1, 1, 3),
   V(0, 1, 3),
   V(1, 0, 3),
   V(0, 0, 1),
```

```
   V(0, 0, 1),
   V(0, 0, 1),
   V(0, 0, 1),


   V(2, 2, 3),
   V(0, 2, 3),
   V(1, 2, 2),
   V(1, 2, 2),
   V(2, 1, 2),
   V(2, 1, 2),
   V(2, 0, 2),
   V(2, 0, 2)
}
```

Definition at line 150 of file huffman.c.

9.14.3.10 union huffpair const hufftab24[ ] [static]

Definition at line 2558 of file huffman.c.

9.14.3.11 union huffpair const hufftab3[ ] [static]

Initial value:

```
 {
   PTR(8, 3),
   V(1, 0, 3),
   V(1, 1, 2),
   V(1, 1, 2),
   V(0, 1, 2),
   V(0, 1, 2),
   V(0, 0, 2),
   V(0, 0, 2),


   V(2, 2, 3),
   V(0, 2, 3),
   V(1, 2, 2),
   V(1, 2, 2),
   V(2, 1, 2),
   V(2, 1, 2),
   V(2, 0, 2),
   V(2, 0, 2)
}
```

Definition at line 172 of file huffman.c.

9.14.3.12 union huffpair const hufftab5[ ] [static]

Definition at line 194 of file huffman.c.

9.14.3.13 union huffpair const hufftab6[ ] [static]

Definition at line 228 of file huffman.c.

9.14.3.14 union huffpair const hufftab7[ ] [static]

Definition at line 266 of file huffman.c.

9.14.3.15 union huffpair const hufftab8[ ] [static]

Definition at line 431 of file huffman.c.

9.14.3.16 union huffpair const hufftab9[ ] [static]

Definition at line 516 of file huffman.c.

9.14.3.17 union huffquad const hufftabA[ ] [static]

Definition at line 57 of file huffman.c.

9.14.3.18 union huffquad const hufftabB[ ] [static]

Initial value:

```
 {
   V(1, 1, 1, 1, 4),
   V(1, 1, 1, 0, 4),
   V(1, 1, 0, 1, 4),
   V(1, 1, 0, 0, 4),
   V(1, 0, 1, 1, 4),
   V(1, 0, 1, 0, 4),
   V(1, 0, 0, 1, 4),
   V(1, 0, 0, 0, 4),
   V(0, 1, 1, 1, 4),
   V(0, 1, 1, 0, 4),
   V(0, 1, 0, 1, 4),
   V(0, 1, 0, 0, 4),
   V(0, 0, 1, 1, 4),
   V(0, 0, 1, 0, 4),
   V(0, 0, 0, 1, 4),
   V(0, 0, 0, 0, 4)
}
```

Definition at line 97 of file huffman.c.

9.14.3.19 struct hufftable const mad_huff_pair_table[32]

Definition at line 3080 of file huffman.c.

Referenced by III_huffdecode().

9.14.3.20 union huffquad const* const mad_huff_quad_table[2] = { hufftabA, hufftabB }

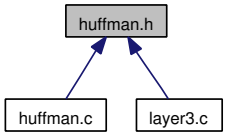
Definition at line 3078 of file huffman.c.

Referenced by III_huffdecode().

9.15 huffman.h File Reference

Declares data structures used in MPEG Layer III Huffman compression decoding.

This graph shows which files directly or indirectly include this file:



Data Structures

- union huffquad
- union huffpair
- struct hufftable

Variables

- union huffquad const ∗const mad_huff_quad_table [2]
- struct hufftable const mad_huff_pair_table [32]

9.15.1 Detailed Description

Declares data structures used in MPEG Layer III Huffman compression decoding.

Definition in file huffman.h.

9.15.2 Variable Documentation

9.15.2.1 struct hufftable const mad_huff_pair_table[32]

Definition at line 3080 of file huffman.c.

Referenced by III_huffdecode().

9.15.2.2 union huffquad const∗ const mad_huff_quad_table[2]

Definition at line 3078 of file huffman.c.

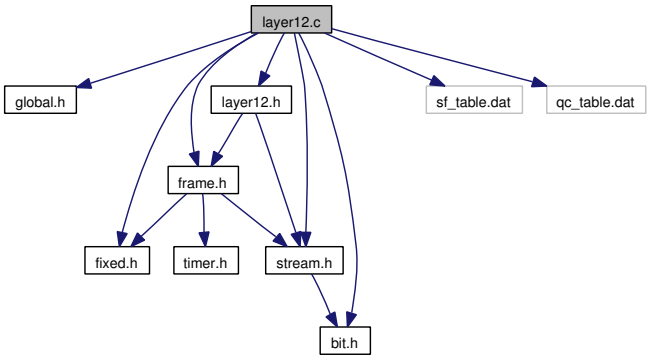
Referenced by III_huffdecode().

9.16 layer12.c File Reference

Implements MPEG Layer I and Layer II methods for mad_stream structure.

```
#include "global.h"
#include "fixed.h"
#include "bit.h"
#include "stream.h"
#include "frame.h"
#include "layer12.h"
#include "sf_table.dat"
#include "qc_table.dat"
```

Include dependency graph for layer12.c:



Data Structures

- struct quantclass

  quantization class table

Defines

- #define CHAR_BIT 8

Functions

- static mad_fixed_t I_sample (struct mad_bitptr *ptr, unsigned int nb)

  Decodes one requantized Layer I sample from a bitstream.

- int mad_layer_I (struct mad_stream *stream, struct mad_frame *frame)

Decodes a single Layer I frame.

- static void II_samples (struct mad_bitptr *ptr, struct quantclass const *quantclass, mad_fixed_t output[3])

  Decodes three requantized Layer II samples from a bitstream.

- int mad_layer_II (struct mad_stream *stream, struct mad_frame *frame)

  Decodes a single Layer II frame.

Variables

- static mad_fixed_t const sf_table [64]

  scalefactor table.

- static mad_fixed_t const linear_table [14]

  linear scaling table

- struct {
  unsigned int sblimit
  unsigned char const offsets [30]
  } sbquant_table [5]

- struct {
  unsigned short nbal
  unsigned short offset
  } bitalloc_table [8]

- static unsigned char const offset_table [6][15]

  offsets into quantization class table

- static struct quantclass qc_table [17]

9.16.1 Detailed Description

Implements MPEG Layer I and Layer II methods for mad_stream structure.

Definition in file layer12.c.

9.16.2 Define Documentation

9.16.2.1 #define CHAR_BIT 8

Definition at line 34 of file layer12.c.

9.16.3 Variable Documentation

9.16.3.1 unsigned short nbal

Definition at line 252 of file layer12.c.

Referenced by mad_layer_II().

9.16.3.2 unsigned short offset

Definition at line 253 of file layer12.c.

9.16.3.3 unsigned char const offsets[30]

Definition at line 232 of file layer12.c.

Referenced by mad_layer_II().

9.16.3.4 unsigned int sblimit

Definition at line 231 of file layer12.c.

Referenced by III_decode(), and mad_layer_II().

9.16.3.5 mad_fixed_t const sf_table[64] [static]

scalefactor table.

used in both Layer I and Layer II decoding

Definition at line 48 of file layer12.c.

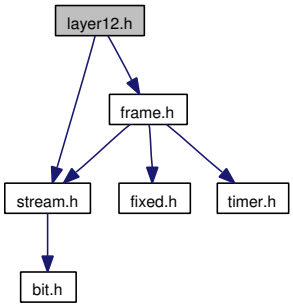
Referenced by mad_layer_I(), and mad_layer_II().

9.17 layer12.h File Reference

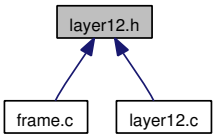
Declares MPEG Layer I and Layer II methods for mad_stream structure.

```
#include "stream.h"
#include "frame.h"
```

Include dependency graph for layer12.h:



This graph shows which files directly or indirectly include this file:



Functions

- int mad_layer_I (struct mad_stream *, struct mad_frame *)
  Decodes a single Layer I frame.
- int mad_layer_II (struct mad_stream *, struct mad_frame *)
  Decodes a single Layer II frame.

9.17.1 Detailed Description

Declares MPEG Layer I and Layer II methods for mad_stream structure.

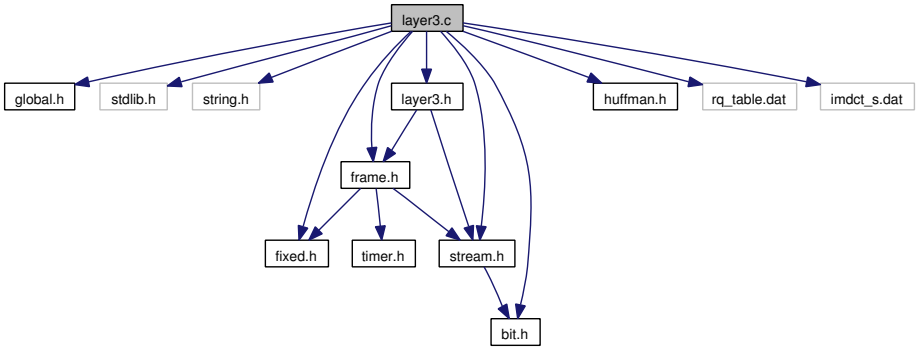
Definition in file layer12.h.

9.18 layer3.c File Reference

Implements MPEG Layer III method for mad_stream structure.

```
#include "global.h"
#include <stdlib.h>
#include <string.h>
#include "fixed.h"
#include "bit.h"
#include "stream.h"
#include "frame.h"
#include "huffman.h"
#include "layer3.h"
#include "rq_table.dat"
#include "imdct_s.dat"
```

Include dependency graph for layer3.c:



Data Structures

- struct sideinfo
- struct sideinfo::granule
- struct sideinfo::granule::channel
- struct fixedfloat

  table for requantization.

Defines

- #define CHAR_BIT 8
- #define sfb_16000_long sfb_22050_long

- #define sfb_12000_long sfb_16000_long

  MPEG 2.5 scalefactor band widths.

- #define sfb_11025_long sfb_12000_long
- #define sfb_12000_short sfb_16000_short
- #define sfb_11025_short sfb_12000_short
- #define sfb_12000_mixed sfb_16000_mixed
- #define sfb_11025_mixed sfb_12000_mixed
- #define MASK(cache, sz, bits) (((cache) >> ((sz) - (bits))) & ((1 << (bits)) - 1))
- #define MASK1BIT(cache, sz) ((cache) & (1 << ((sz) - 1)))

Enumerations

- enum { count1table_select = 0x01, scalefac_scale = 0x02, preflag = 0x04, mixed_block_flag = 0x08 }
- enum { I_STEREO = 0x1, MS_STEREO = 0x2 }

Functions

- static enum mad_error III_sideinfo (struct mad_bitptr *ptr, unsigned int nch, int lsf, struct sideinfo *si, unsigned int *data_bitlen, unsigned int *priv_bitlen)

  decode frame side information from a bitstream.

- static unsigned int III_scalefactors_lsf (struct mad_bitptr *ptr, struct channel *channel, struct channel *gr1ch, int mode_extension)

  decode channel scalefactors for LSF from a bitstream.

- static unsigned int III_scalefactors (struct mad_bitptr *ptr, struct channel *channel, struct channel const *gr0ch, unsigned int scfsi)

  decode channel scalefactors of one granule from a bitstream.

- static void III_exponents (struct channel const *channel, unsigned char const *sfbwidth, signed int exponents[39])

  The Layer III formula for requantization and scaling is defined by section 2.4.3.4.7.1 of ISO/IEC 11172-3, as follows:.

- static mad_fixed_t III_requantize (unsigned int value, signed int exp)

  requantize one (positive) value.

- static enum mad_error III_huffdecode (struct mad_bitptr *ptr, mad_fixed_t xr[576], struct channel *channel, unsigned char const *sfbwidth, unsigned int part2_length)

  decode Huffman code words of one channel of one granule.

- static void III_reorder (mad_fixed_t xr[576], struct channel const *channel, unsigned char const sfbwidth[39])

  reorder frequency lines of a short block into subband order.

- static enum mad_error III_stereo (mad_fixed_t xr[2][576], struct granule const *granule, struct mad_header *header, unsigned char const *sfbwidth)

  perform joint stereo processing on a granule.

- static void III_aliasreduce (mad_fixed_t xr[576], int lines)

  perform frequency line alias reduction.

- static void fastsdct (mad_fixed_t const x[9], mad_fixed_t y[18])
- static void sdctII (mad_fixed_t const x[18], mad_fixed_t X[18])
- static void dctIV (mad_fixed_t const y[18], mad_fixed_t X[18])
- static void imdct36 (mad_fixed_t const x[18], mad_fixed_t y[36])

  perform X[18]->x[36] IMDCT using Szu-Wei Lee's fast algorithm.

- static void III_imdct_l (mad_fixed_t const X[18], mad_fixed_t z[36], unsigned int block_type)

  perform IMDCT and windowing for long blocks.

- static void III_imdct_s (mad_fixed_t const X[18], mad_fixed_t z[36])

  perform IMDCT and windowing for short blocks.

- static void III_overlap (mad_fixed_t const output[36], mad_fixed_t overlap[18], mad_fixed_t sample[18][32], unsigned int sb)

  perform overlap-add of windowed IMDCT outputs.

- static void III_overlap_z (mad_fixed_t overlap[18], mad_fixed_t sample[18][32], unsigned int sb)

  perform "overlap-add" of zero IMDCT outputs.

- static void III_freqinver (mad_fixed_t sample[18][32], unsigned int sb)

  perform subband frequency inversion for odd sample lines.

- static enum mad_error III_decode (struct mad_bitptr *ptr, struct mad_frame *frame, struct sideinfo *si, unsigned int nch)

  decode frame main_data.

- int mad_layer_III (struct mad_stream *stream, struct mad_frame *frame)

  decode a single Layer III frame.

Variables

- struct {
  unsigned char slen1
  unsigned char slen2
  } sflen_table [16]

- static unsigned char const nsfb_table [6][3][4]

  number of LSF scalefactor band values.

- static unsigned char const sfb_48000_long [ ]

  MPEG-1 scalefactor band widths.

- static unsigned char const sfb_44100_long [ ]
- static unsigned char const sfb_32000_long [ ]
- static unsigned char const sfb_48000_short [ ]
- static unsigned char const sfb_44100_short [ ]
- static unsigned char const sfb_32000_short [ ]
- static unsigned char const sfb_48000_mixed [ ]
- static unsigned char const sfb_44100_mixed [ ]
- static unsigned char const sfb_32000_mixed [ ]
- static unsigned char const sfb_24000_long [ ]

  MPEG-2 scalefactor band widths.

- static unsigned char const sfb_22050_long [ ]
- static unsigned char const sfb_24000_short [ ]
- static unsigned char const sfb_22050_short [ ]
- static unsigned char const sfb_16000_short [ ]
- static unsigned char const sfb_24000_mixed [ ]
- static unsigned char const sfb_22050_mixed [ ]
- static unsigned char const sfb_16000_mixed [ ]
- static unsigned char const sfb_8000_long [ ]
- static unsigned char const sfb_8000_short [ ]
- static unsigned char const sfb_8000_mixed [ ]
- struct {
  unsigned char const * l
  unsigned char const * s
  unsigned char const * m
  } sfbwidth_table [9]

- static unsigned char const pretab [22]

  scalefactor band preemphasis (used only when preflag is set).

- static struct fixedfloat rq_table [8207]
- static mad_fixed_t const root_table [7]

  fractional powers of two.

- static mad_fixed_t const cs [8]
  coefficients for aliasing reduction.
- static mad_fixed_t const ca [8]
- static mad_fixed_t const imdct_s [6][6]
  IMDCT coefficients for short blocks.
- static mad_fixed_t const window_l [36]
  windowing coefficients for long blocks.
- static mad_fixed_t const window_s [12]
  windowing coefficients for short blocks.
- static mad_fixed_t const is_table [7]
  coefficients for intensity stereo processing.
- static mad_fixed_t const is_lsf_table [2][15]
  coefficients for LSF intensity stereo processing.

9.18.1 Detailed Description

Implements MPEG Layer III method for mad_stream structure.

Definition in file layer3.c.

9.18.2 Define Documentation

9.18.2.1 #define CHAR_BIT 8

Definition at line 41 of file layer3.c.

9.18.3 Variable Documentation

9.18.3.1 unsigned char const* l

Definition at line 308 of file layer3.c.

Referenced by III_decode(), III_exponents(), III_reorder(), and III_stereo().

9.18.3.2 unsigned char const* m

Definition at line 310 of file layer3.c.

Referenced by III_decode(), III_stereo(), and mad_timer_string().

9.18.3.3 unsigned char const* s

Definition at line 309 of file layer3.c.

Referenced by II_samples(), III_decode(), III_imdct_s(), III_stereo(), mad_frame_mute(), mad_layer_I(), mad_layer_II(), mad_synth_mute(), synth_full(), and synth_half().

9.18.3.4 unsigned char slen1

Definition at line 100 of file layer3.c.

Referenced by III_scalefactors().

9.18.3.5 unsigned char slen2

Definition at line 101 of file layer3.c.

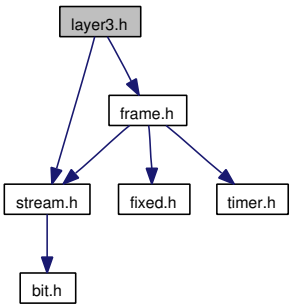
Referenced by III_scalefactors().

9.19 layer3.h File Reference

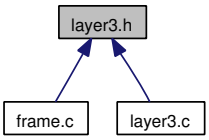
Declares MPEG Layer III method for mad_stream structure.

```
#include "stream.h"
#include "frame.h"
```

Include dependency graph for layer3.h:



This graph shows which files directly or indirectly include this file:



Functions

- int mad_layer_III (struct mad_stream *, struct mad_frame *)
  decode a single Layer III frame.

9.19.1 Detailed Description

Declares MPEG Layer III method for mad_stream structure.

Definition in file layer3.h.

9.20 libmad.dox File Reference

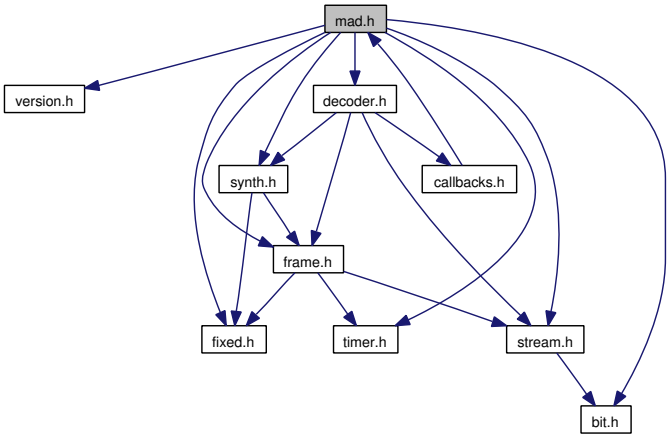
9.21 lowlevel.dox File Reference

9.22 mad.h File Reference

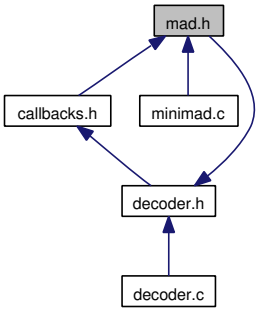
Includes most usual libmad header files.

```
#include "version.h"
#include "fixed.h"
#include "bit.h"
#include "timer.h"
#include "stream.h"
#include "frame.h"
#include "synth.h"
#include "decoder.h"
```

Include dependency graph for mad.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define FPM_INTEL
- #define SIZEOF_INT 4
- #define SIZEOF_LONG 4
- #define SIZEOF_LONG_LONG 8

9.22.1 Detailed Description

Includes most usual libmad header files.

Definition in file mad.h.

9.22.2 Define Documentation

9.22.2.1 #define FPM_INTEL

Definition at line 30 of file mad.h.

9.22.2.2 #define SIZEOF_INT 4

Definition at line 34 of file mad.h.

9.22.2.3 #define SIZEOF_LONG 4

Definition at line 35 of file mad.h.

9.22.2.4 #define SIZEOF_LONG_LONG 8

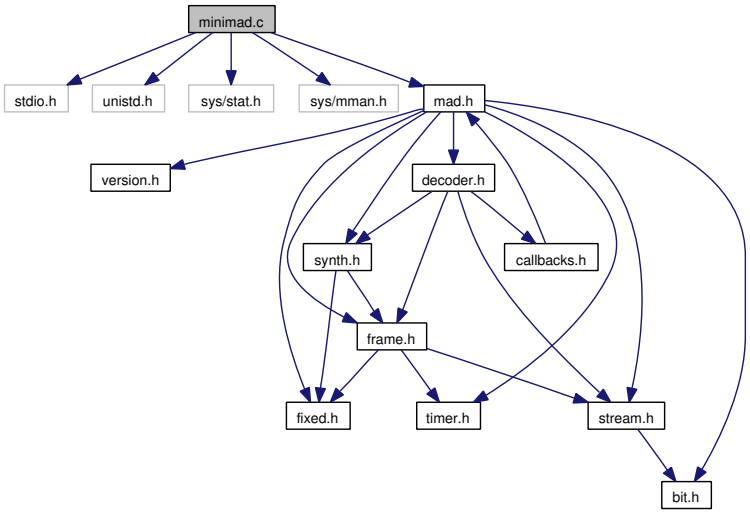
Definition at line 36 of file mad.h.

9.23 minimad.c File Reference

This is perhaps the simplest example use of the MAD high-level API.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include "mad.h"
```

Include dependency graph for minimad.c:



Data Structures

- struct buffer

  This is a private message structure.

Functions

- static int decode (unsigned char const *start, unsigned long length)

  This is the function called by main() above to perform all the decoding.

- int main (int argc, char *argv[ ])
- static enum mad_flow input (void *data, struct mad_stream *stream)

  This is the input callback.

- static signed int scale (mad_fixed_t sample)

The following utility routine performs simple rounding, clipping, and scaling of MAD's high-resolution samples down to 16 bits.

- static enum mad_flow output (void *data, struct mad_header const *header, struct mad_pcm *pcm)

  This is the output callback function.

- static enum mad_flow error (void *data, struct mad_stream *stream, struct mad_frame *frame)

  This is the error callback function.

9.23.1 Detailed Description

This is perhaps the simplest example use of the MAD high-level API.

Standard input is mapped into memory via mmap(), then the high-level API is invoked with three callbacks: input, output, and error. The output callback converts MAD's high-resolution PCM samples to 16 bits, then writes them to standard output in little-endian, stereo-interleaved format.

See also:

main()

Definition in file minimad.c.

9.23.2 Function Documentation

9.23.2.1 static int decode (unsigned char const * *start*, unsigned long *length*) [static]

This is the function called by main() above to perform all the decoding.

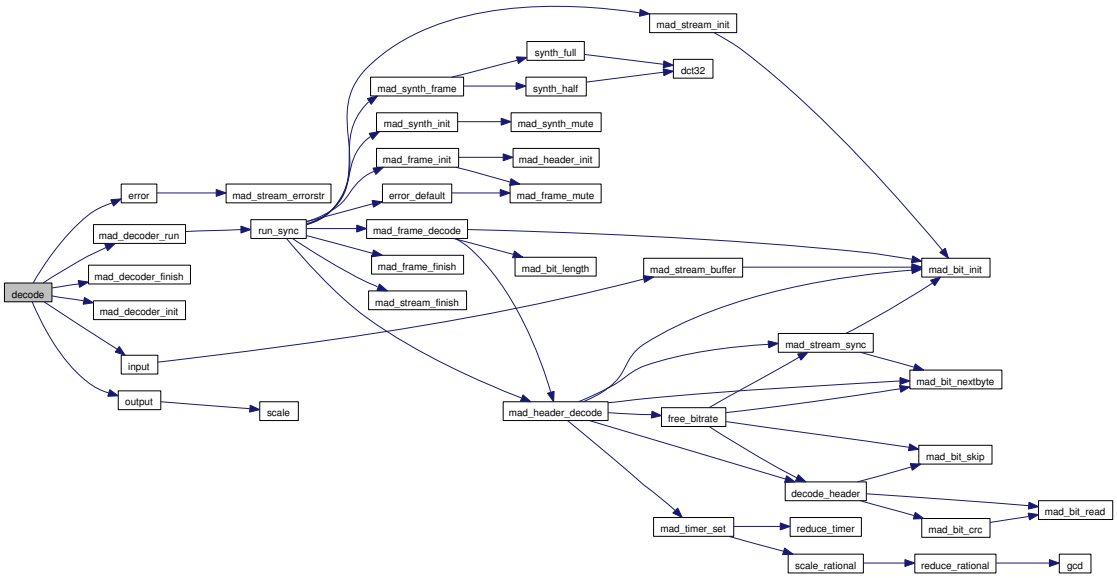
It instantiates a decoder object and configures it with the input, output, and error callback functions above. A single call to mad_decoder_run() continues until a callback function returns MAD_FLOW_STOP (to stop decoding) or MAD_FLOW_BREAK (to stop decoding and signal an error).

Definition at line 198 of file minimad.c.

References error(), input(), buffer::length, mad_decoder_finish(), mad_decoder_init(), MAD_DECODER_MODE_SYNC, mad_decoder_run(), output(), and buffer::start.

Referenced by main().

Here is the call graph for this function:



9.23.2.2 static enum mad_flow error (void * *data*, struct mad_stream * *stream*, struct mad_frame * *frame*) [static]

This is the error callback function.

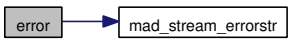
It is called whenever a decoding error occurs. The error is indicated by stream->error; the list of possible MAD_ERROR_* errors can be found in the mad.h (or stream.h) header file.

Definition at line 173 of file minimad.c.

References mad_stream::error, MAD_FLOW_CONTINUE, mad_stream_errorstr(), buffer::start, and mad_stream::this_frame.

Referenced by decode(), III_decode(), and mad_layer_III().

Here is the call graph for this function:



9.23.2.3 static enum mad_flow input (void * *data*, struct mad_stream * *stream*) [static]

This is the input callback.

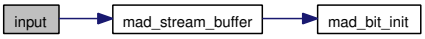
The purpose of this callback is to (re)fill the stream buffer which is to be decoded. In this example, an entire file has been mapped into memory, so we just call mad_stream_buffer() with the address and length of the mapping. When this callback is called a second time, we are finished decoding.

Definition at line 86 of file minimad.c.

References buffer::length, MAD_FLOW_CONTINUE, MAD_FLOW_STOP, mad_stream_buffer(), and buffer::start.

Referenced by decode().

Here is the call graph for this function:

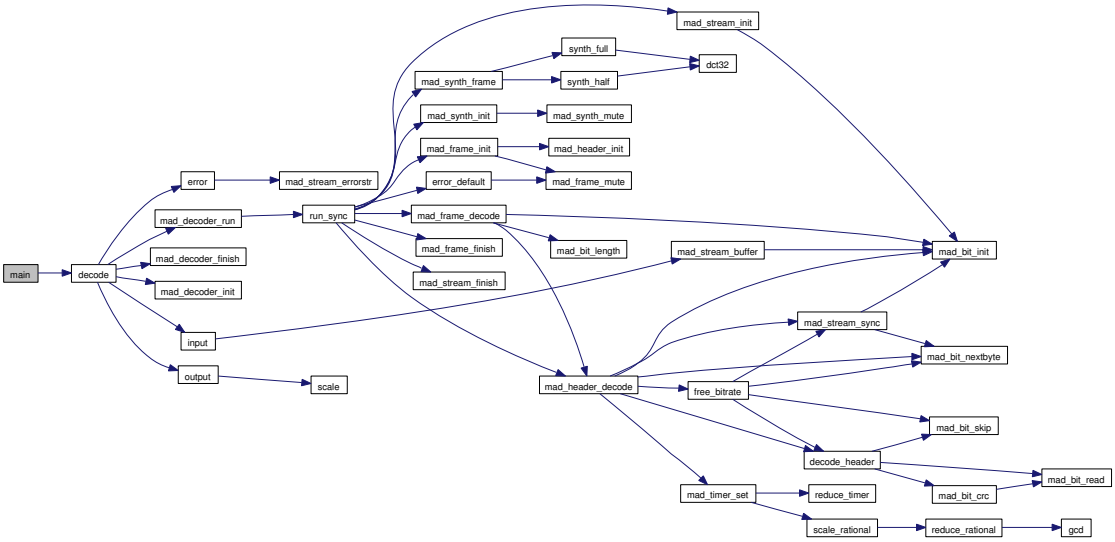


9.23.2.4 int main (int *argc*, char * *argv*[ ])

Definition at line 42 of file minimad.c.

References decode().

Here is the call graph for this function:



9.23.2.5 static enum mad_flow output (void * *data*, struct mad_header const * *header*, struct mad_pcm * *pcm*) [static]

This is the output callback function.

It is called after each frame of MPEG audio data has been completely decoded. The purpose of this callback is to output (or play) the decoded PCM audio.

Definition at line 132 of file minimad.c.

References mad_pcm::channels, mad_pcm::length, MAD_FLOW_CONTINUE, mad_pcm::samples, and scale().

Referenced by decode(), and III_decode().

Here is the call graph for this function:

9.23.2.6 static signed int scale (mad_fixed_t *sample*) [inline, static]

The following utility routine performs simple rounding, clipping, and scaling of MAD's high-resolution samples down to 16 bits.

It does not perform any dithering or noise shaping, which would be recommended to obtain any exceptional audio quality. It is therefore not recommended to use this routine if high-quality output is desired.

Definition at line 110 of file minimad.c.

References MAD_F_FRACBITS, and MAD_F_ONE.

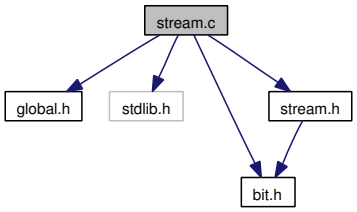
Referenced by dctIV(), output(), and sdctII().

9.24 stream.c File Reference

Implementation of the mad_stream methods, which handle the input stream buffer which is to be decoded.

```
#include "global.h"
#include <stdlib.h>
#include "bit.h"
#include "stream.h"
```

Include dependency graph for stream.c:



Functions

- void mad_stream_init (struct mad_stream *stream)
  Initializes stream struct.
- void mad_stream_finish (struct mad_stream *stream)
  Deallocates any dynamic memory associated with stream.
- void mad_stream_buffer (struct mad_stream *stream, unsigned char const *buffer, unsigned long length)
  Sets stream buffer pointers.
- void mad_stream_skip (struct mad_stream *stream, unsigned long length)
  Arranges to skip bytes before the next frame.
- int mad_stream_sync (struct mad_stream *stream)
  Locates the next stream sync word.
- char const * mad_stream_errorstr (struct mad_stream const *stream)
  Returns a string description of the current error condition.

9.24.1 Detailed Description

Implementation of the mad_stream methods, which handle the input stream buffer which is to be decoded.

Definition in file stream.c.

9.24.2 Function Documentation

9.24.2.1 void mad_stream_buffer (struct mad_stream * *stream*, unsigned char const * *buffer*, unsigned long *length*)

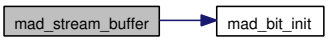
Sets stream buffer pointers.

Definition at line 80 of file stream.c.

References mad_stream::bufend, mad_stream::buffer, mad_bit_init(), mad_stream::next_frame, mad_stream::ptr, mad_stream::sync, and mad_stream::this_frame.

Referenced by input().

Here is the call graph for this function:



9.24.2.2 char const* mad_stream_errorstr (struct mad_stream const * *stream*)

Returns a string description of the current error condition.

Definition at line 127 of file stream.c.

References mad_stream::error, MAD_ERROR_BADBIGVALUES, MAD_ERROR_BADBITALLOC, MAD_ERROR_BADBITRATE, MAD_ERROR_BADBLOCKTYPE, MAD_ERROR_BADCRC, MAD_ERROR_BADDATAPTR, MAD_ERROR_BADEMPHASIS, MAD_ERROR_BADFRAMELEN, MAD_ERROR_BADHUFFDATA, MAD_ERROR_BADHUFFTABLE, MAD_ERROR_BADLAYER, MAD_ERROR_BADMODE, MAD_ERROR_BADPART3LEN, MAD_ERROR_BADSAMPLERATE, MAD_ERROR_BADSCALEFACTOR, MAD_ERROR_BADSCFSI, MAD_ERROR_BADSTEREO, MAD_ERROR_BUFLEN, MAD_ERROR_BUFPTR, MAD_ERROR_LOSTSYNC, MAD_ERROR_NOMEM, and MAD_ERROR_NONE.

Referenced by error().

9.24.2.3 void mad_stream_finish (struct mad_stream * *stream*)

Deallocates any dynamic memory associated with stream.

Definition at line 66 of file stream.c.

References mad_stream::anc_ptr, mad_bit_finish, mad_stream::main_data, and mad_stream::ptr.

Referenced by run_sync().

9.24.2.4 void mad_stream_init (struct mad_stream * *stream*)

Initializes stream struct.

Definition at line 40 of file stream.c.

References mad_stream::anc_bitlen, mad_stream::anc_ptr, mad_stream::bufend, mad_stream::buffer, mad_stream::error, mad_stream::freerate, mad_bit_init(), MAD_ERROR_NONE, mad_stream::main_data, mad_stream::md_len, mad_stream::next_frame, mad_stream::options, mad_stream::ptr, mad_stream::skiplen, mad_stream::sync, and mad_stream::this_frame.

Referenced by run_sync().

Here is the call graph for this function:

9.24.2.5 void mad_stream_skip (struct mad_stream * *stream*, unsigned long *length*)

Arranges to skip bytes before the next frame.

Definition at line 97 of file stream.c.

References mad_stream::skiplen.

9.24.2.6 int mad_stream_sync (struct mad_stream * *stream*)

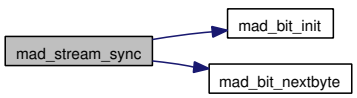
Locates the next stream sync word.

Definition at line 105 of file stream.c.

References mad_stream::bufend, mad_bit_init(), mad_bit_nextbyte(), MAD_BUFFER_GUARD, and mad_stream::ptr.

Referenced by free_bitrate(), and mad_header_decode().

Here is the call graph for this function:

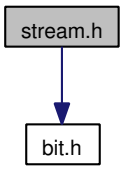


9.25 stream.h File Reference

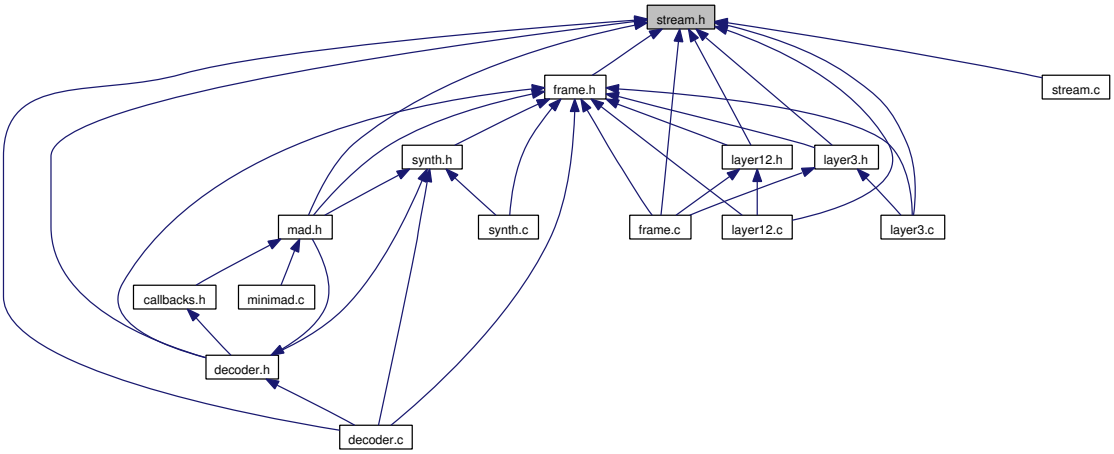
Defines the mad_stream structure and methods, which handle the input stream buffer which is to be decoded.

```
#include "bit.h"
```

Include dependency graph for stream.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct mad_stream

Defines

- #define MAD_BUFFER_GUARD 8
- #define MAD_BUFFER_MDLEN (511 + 2048 + MAD_BUFFER_GUARD)
- #define MAD_RECOVERABLE(error) ((error) & 0xff00)
- #define mad_stream_options(stream, opts) ((void) ((stream) → options = (opts)))

Enumerations

- enum mad_error {

  MAD_ERROR_NONE = 0x0000, MAD_ERROR_BUFLEN = 0x0001, MAD_ERROR_BUFPTR = 0x0002, MAD_ERROR_NOMEM = 0x0031,

  MAD_ERROR_LOSTSYNC = 0x0101, MAD_ERROR_BADLAYER = 0x0102, MAD_ERROR_BADBITRATE = 0x0103, MAD_ERROR_BADSAMPLERATE = 0x0104,

  MAD_ERROR_BADEMPHASIS = 0x0105, MAD_ERROR_BADCRC = 0x0201, MAD_ERROR_BADBITALLOC = 0x0211, MAD_ERROR_BADSCALEFACTOR = 0x0221,

  MAD_ERROR_BADMODE = 0x0222, MAD_ERROR_BADFRAMELEN = 0x0231, MAD_ERROR_BADBIGVALUES = 0x0232, MAD_ERROR_BADBLOCKTYPE = 0x0233,

  MAD_ERROR_BADSCFSI = 0x0234, MAD_ERROR_BADDATAPTR = 0x0235, MAD_ERROR_BADPART3LEN = 0x0236, MAD_ERROR_BADHUFFTABLE = 0x0237,

  MAD_ERROR_BADHUFFDATA = 0x0238, MAD_ERROR_BADSTEREO = 0x0239 }
- enum { MAD_OPTION_IGNORECRC = 0x0001, MAD_OPTION_HALFSAMPLERATE = 0x0002 }

Functions

- void mad_stream_init (struct mad_stream *)

  Initializes stream struct.

- void mad_stream_finish (struct mad_stream *)

  Deallocates any dynamic memory associated with stream.

- void mad_stream_buffer (struct mad_stream *, unsigned char const *, unsigned long)

  Sets stream buffer pointers.

- void mad_stream_skip (struct mad_stream *, unsigned long)

  Arranges to skip bytes before the next frame.

- int mad_stream_sync (struct mad_stream *)

  Locates the next stream sync word.

- char const * mad_stream_errorstr (struct mad_stream const *)

  Returns a string description of the current error condition.

9.25.1 Detailed Description

Defines the mad_stream structure and methods, which handle the input stream buffer which is to be decoded.

Definition in file stream.h.

9.25.2 Define Documentation

9.25.2.1 #define MAD_BUFFER_GUARD 8

Definition at line 31 of file stream.h.

Referenced by III_huffdecode(), mad_header_decode(), and mad_stream_sync().

9.25.2.2 #define MAD_BUFFER_MDLEN (511 + 2048 + MAD_BUFFER_GUARD)

Definition at line 32 of file stream.h.

Referenced by mad_layer_III().

9.25.2.3 #define MAD_RECOVERABLE(error) ((error) & 0xff00)

Definition at line 63 of file stream.h.

Referenced by mad_frame_decode(), and run_sync().

9.25.2.4 #define mad_stream_options(stream, opts) ((void) ((stream) → options = (opts)))

Definition at line 101 of file stream.h.

Referenced by run_sync().

9.25.3 Enumeration Type Documentation

9.25.3.1 anonymous enum

Enumerator:

MAD_OPTION_IGNORECRC ignore CRC errors

MAD_OPTION_HALFSAMPLERATE generate PCM at 1/2 sample rate

Definition at line 88 of file stream.h.

9.25.3.2 enum mad_error

Enumerator:

MAD_ERROR_NONE no error

MAD_ERROR_BUFLEN input buffer too small (or EOF)

MAD_ERROR_BUFPTR invalid (null) buffer pointer

MAD_ERROR_NOMEM not enough memory

MAD_ERROR_LOSTSYNC lost synchronization

MAD_ERROR_BADLAYER reserved header layer value

MAD_ERROR_BADBITRATE forbidden bitrate value

MAD_ERROR_BADSAMPLERATE reserved sample frequency value

MAD_ERROR_BADEMPHASIS reserved emphasis value

MAD_ERROR_BADCRC CRC check failed.

MAD_ERROR_BADBITALLOC forbidden bit allocation value

MAD_ERROR_BADSCALEFACTOR bad scalefactor index

MAD_ERROR_BADMODE bad bitrate/mode combination

MAD_ERROR_BADFRAMELEN bad frame length

MAD_ERROR_BADBIGVALUES bad big_values count

MAD_ERROR_BADBLOCKTYPE reserved block_type

MAD_ERROR_BADSCFSI bad scalefactor selection info

MAD_ERROR_BADDATAPTR bad main_data_begin pointer

MAD_ERROR_BADPART3LEN bad audio data length

MAD_ERROR_BADHUFFTABLE bad Huffman table select

MAD_ERROR_BADHUFFDATA Huffman data overrun.

MAD_ERROR_BADSTEREO incompatible block_type for JS

Definition at line 34 of file stream.h.

9.25.4 Function Documentation

9.25.4.1 void mad_stream_buffer (struct mad_stream *, unsigned char const *, unsigned *long*)

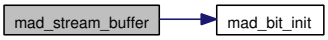
Sets stream buffer pointers.

Definition at line 80 of file stream.c.

References mad_stream::bufend, mad_stream::buffer, mad_bit_init(), mad_stream::next_frame, mad_stream::ptr, mad_stream::sync, and mad_stream::this_frame.

Referenced by input().

Here is the call graph for this function:



9.25.4.2 char const* mad_stream_errorstr (struct mad_stream const *)

Returns a string description of the current error condition.

Definition at line 127 of file stream.c.

References mad_stream::error, MAD_ERROR_BADBIGVALUES, MAD_ERROR_BADBITALLOC, MAD_ERROR_BADBITRATE, MAD_ERROR_BADBLOCKTYPE, MAD_ERROR_BADCRC, MAD_ERROR_BADDATAPTR, MAD_ERROR_BADEMPHASIS, MAD_ERROR_BADFRAMELEN, MAD_ERROR_BADHUFFDATA, MAD_ERROR_BADHUFFTABLE, MAD_ERROR_BADLAYER, MAD_ERROR_BADMODE, MAD_ERROR_BADPART3LEN, MAD_ERROR_BADSAMPLERATE, MAD_ERROR_BADSCALEFACTOR, MAD_ERROR_BADSCFSI, MAD_ERROR_BADSTEREO, MAD_ERROR_BUFLEN, MAD_ERROR_BUFPTR, MAD_ERROR_LOSTSYNC, MAD_ERROR_NOMEM, and MAD_ERROR_NONE.

Referenced by error().

9.25.4.3 void mad_stream_finish (struct mad_stream *)

Deallocates any dynamic memory associated with stream.

Definition at line 66 of file stream.c.

References mad_stream::anc_ptr, mad_bit_finish, mad_stream::main_data, and mad_stream::ptr.

Referenced by run_sync().

9.25.4.4 void mad_stream_init (struct mad_stream *)

Initializes stream struct.

Definition at line 40 of file stream.c.

References mad_stream::anc_bitlen, mad_stream::anc_ptr, mad_stream::bufend, mad_stream::buffer, mad_stream::error, mad_stream::freerate, mad_bit_init(), MAD_ERROR_NONE, mad_stream::main_data, mad_stream::md_len, mad_stream::next_frame, mad_stream::options, mad_stream::ptr, mad_stream::skiplen, mad_stream::sync, and mad_stream::this_frame.

Referenced by run_sync().

Here is the call graph for this function:

9.25.4.5 void mad_stream_skip (struct mad_stream *, unsigned *long*)

Arranges to skip bytes before the next frame.

Definition at line 97 of file stream.c.

References mad_stream::skiplen.

9.25.4.6 int mad_stream_sync (struct mad_stream *)

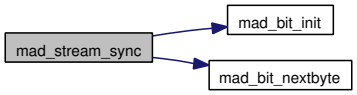
Locates the next stream sync word.

Definition at line 105 of file stream.c.

References mad_stream::bufend, mad_bit_init(), mad_bit_nextbyte(), MAD_-BUFFER_GUARD, and mad_stream::ptr.

Referenced by free_bitrate(), and mad_header_decode().

Here is the call graph for this function:

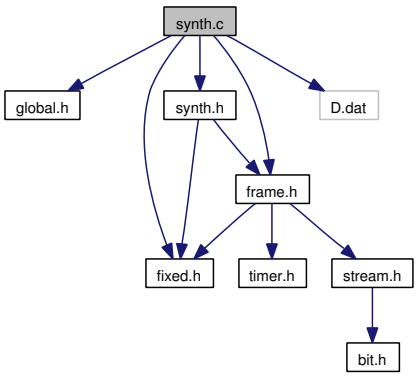


9.26 synth.c File Reference

Implements the mad_synth methods, to perform PCM synthesis.

```
#include "global.h"
#include "fixed.h"
#include "frame.h"
#include "synth.h"
#include "D.dat"
```

Include dependency graph for synth.c:



Defines

- #define SHIFT(x) (x)

  FPM_DEFAULT without OPT_SSO will actually lose accuracy and performance.

- #define MUL(x, y) mad_f_mul((x), (y))
- #define costab1 MAD_F(0x0ffb10f2)
- #define costab2 MAD_F(0x0fec46d2)
- #define costab3 MAD_F(0x0fd3aac0)
- #define costab4 MAD_F(0x0fb14be8)
- #define costab5 MAD_F(0x0f853f7e)
- #define costab6 MAD_F(0x0f4fa0ab)
- #define costab7 MAD_F(0x0f109082)
- #define costab8 MAD_F(0x0ec835e8)
- #define costab9 MAD_F(0x0e76bd7a)
- #define costab10 MAD_F(0x0e1c5979)
- #define costab11 MAD_F(0x0db941a3)
- #define costab12 MAD_F(0x0d4db315)
- #define costab13 MAD_F(0x0cd9f024)
- #define costab14 MAD_F(0x0c5e4036)
- #define costab15 MAD_F(0x0bdaef91)

- #define costab16 MAD_F(0x0b504f33)
- #define costab17 MAD_F(0x0abeb49a)
- #define costab18 MAD_F(0x0a267993)
- #define costab19 MAD_F(0x0987fbfe)
- #define costab20 MAD_F(0x08e39d9d)
- #define costab21 MAD_F(0x0839c3cd)
- #define costab22 MAD_F(0x078ad74e)
- #define costab23 MAD_F(0x06d74402)
- #define costab24 MAD_F(0x061f78aa)
- #define costab25 MAD_F(0x0563e69d)
- #define costab26 MAD_F(0x04a5018c)
- #define costab27 MAD_F(0x03e33f2f)
- #define costab28 MAD_F(0x031f1708)
- #define costab29 MAD_F(0x0259020e)
- #define costab30 MAD_F(0x01917a6c)
- #define costab31 MAD_F(0x00c8fb30)
- #define ML0(hi, lo, x, y) MAD_F_ML0((hi), (lo), (x), (y))

  third SSO shift and/or D[] optimization preshift

- #define MLA(hi, lo, x, y) MAD_F_MLA((hi), (lo), (x), (y))
- #define MLN(hi, lo) MAD_F_MLN((hi), (lo))
- #define MLZ(hi, lo) MAD_F_MLZ((hi), (lo))
- #define SHIFT(x) (x)

  FPM_DEFAULT without OPT_SSO will actually lose accuracy and performance.

- #define MAD_F_SCALEBITS (MAD_F_FRACBITS - 12)
- #define PRESHIFT(x) (MAD_F(x) >> 12)

Functions

- void mad_synth_init (struct mad_synth *synth)

  Initializes synth struct.

- void mad_synth_mute (struct mad_synth *synth)

  Zeroes all polyphase filterbank values, resetting synthesis.

- static void dct32 (mad_fixed_t const in[32], unsigned int slot, mad_fixed_t lo[16][8], mad_fixed_t hi[16][8])

  Performs fast in[32]->out[32] DCT.

- static void synth_full (struct mad_synth *synth, struct mad_frame const *frame, unsigned int nch, unsigned int ns)

  perform full frequency PCM synthesis.

- static void synth_half (struct mad_synth *synth, struct mad_frame const *frame, unsigned int nch, unsigned int ns)

perform half frequency PCM synthesis.

- void mad_synth_frame (struct mad_synth *synth, struct mad_frame const *frame)

  perform PCM synthesis of frame subband samples.

Variables

- static mad_fixed_t const D [17][32]

9.26.1 Detailed Description

Implements the mad_synth methods, to perform PCM synthesis.

Definition in file synth.c.

9.26.2 Define Documentation

9.26.2.1 #define costab1 MAD_F(0x0ffb10f2)

Referenced by dct32().

9.26.2.2 #define costab10 MAD_F(0x0e1c5979)

Referenced by dct32().

9.26.2.3 #define costab11 MAD_F(0x0db941a3)

Referenced by dct32().

9.26.2.4 #define costab12 MAD_F(0x0d4db315)

Referenced by dct32().

9.26.2.5 #define costab13 MAD_F(0x0cd9f024)

Referenced by dct32().

9.26.2.6 #define costab14 MAD_F(0x0c5e4036)

Referenced by dct32().

9.26.2.7 #define costab15 MAD_F(0x0bdaef91)

Referenced by dct32().

9.26.2.8 #define costab16 MAD_F(0x0b504f33)

Referenced by dct32().

9.26.2.9 #define costab17 MAD_F(0x0abeb49a)

Referenced by dct32().

9.26.2.10 #define costab18 MAD_F(0x0a267993)

Referenced by dct32().

9.26.2.11 #define costab19 MAD_F(0x0987fbfe)

Referenced by dct32().

9.26.2.12 #define costab2 MAD_F(0x0fec46d2)

Referenced by dct32().

9.26.2.13 #define costab20 MAD_F(0x08e39d9d)

Referenced by dct32().

9.26.2.14 #define costab21 MAD_F(0x0839c3cd)

Referenced by dct32().

9.26.2.15 #define costab22 MAD_F(0x078ad74e)

Referenced by dct32().

9.26.2.16 #define costab23 MAD_F(0x06d74402)

Referenced by dct32().

9.26.2.17 #define costab24 MAD_F(0x061f78aa)

Referenced by dct32().

9.26.2.18 #define costab25 MAD_F(0x0563e69d)

Referenced by dct32().

9.26.2.19 #define costab26 MAD_F(0x04a5018c)

Referenced by dct32().

9.26.2.20 #define costab27 MAD_F(0x03e33f2f)

Referenced by dct32().

9.26.2.21 #define costab28 MAD_F(0x031f1708)

Referenced by dct32().

9.26.2.22 #define costab29 MAD_F(0x0259020e)

Referenced by dct32().

9.26.2.23 #define costab3 MAD_F(0x0fd3aac0)

Referenced by dct32().

9.26.2.24 #define costab30 MAD_F(0x01917a6c)

Referenced by dct32().

9.26.2.25 #define costab31 MAD_F(0x00c8fb30)

Referenced by dct32().

9.26.2.26 #define costab4 MAD_F(0x0fb14be8)

Referenced by dct32().

9.26.2.27 #define costab5 MAD_F(0x0f853f7e)

Referenced by dct32().

9.26.2.28 #define costab6 MAD_F(0x0f4fa0ab)

Referenced by dct32().

9.26.2.29 #define costab7 MAD_F(0x0f109082)

Referenced by dct32().

9.26.2.30 #define costab8 MAD_F(0x0ec835e8)

Referenced by dct32().

9.26.2.31 #define costab9 MAD_F(0x0e76bd7a)

Referenced by dct32().

9.26.2.32 #define MAD_F_SCALEBITS (MAD_F_FRACBITS - 12)

Definition at line 543 of file synth.c.

9.26.2.33 #define ML0(hi, lo, x, y) MAD_F_ML0((hi), (lo), (x), (y))

third SSO shift and/or D[] optimization preshift

Definition at line 536 of file synth.c.

Referenced by synth_full(), and synth_half().

9.26.2.34 #define MLA(hi, lo, x, y) MAD_F_MLA((hi), (lo), (x), (y))

Definition at line 537 of file synth.c.

Referenced by synth_full(), and synth_half().

9.26.2.35 #define MLN(hi, lo) MAD_F_MLN((hi), (lo))

Definition at line 538 of file synth.c.

Referenced by synth_full(), and synth_half().

9.26.2.36 #define MLZ(hi, lo) MAD_F_MLZ((hi), (lo))

Definition at line 539 of file synth.c.

Referenced by synth_full(), and synth_half().

9.26.2.37 #define PRESHIFT(x) (MAD_F(x) >> 12)

Definition at line 544 of file synth.c.

9.26.3 Function Documentation

9.26.3.1 static void dct32 (mad_fixed_t const *in*[32], unsigned int *slot*, mad_fixed_t *lo*[16][8], mad_fixed_t *hi*[16][8]) [static]

Performs fast in[32]->out[32] DCT.

costab[i] = cos(PI / (2 * 32) * i)

Definition at line 129 of file synth.c.

References costab1, costab10, costab11, costab12, costab13, costab14, costab15, costab16, costab17, costab18, costab19, costab2, costab20, costab21, costab22, costab23, costab24, costab25, costab26, costab27, costab28, costab29, costab3, costab30, costab31, costab4, costab5, costab6, costab7, costab8, costab9, MUL, and SHIFT.

Referenced by synth_full(), and synth_half().

9.26.3.2 void mad_synth_frame (struct mad_synth * *synth*, struct mad_frame const * *frame*)

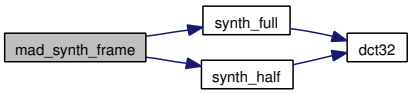
perform PCM synthesis of frame subband samples.

Definition at line 835 of file synth.c.

References mad_pcm::channels, mad_frame::header, mad_pcm::length, MAD_NCHANNELS, MAD_NSBSAMPLES, MAD_OPTION_HALFSAMPLERATE, mad_frame::options, mad_synth::pcm, mad_synth::phase, mad_header::samplerate, mad_pcm::samplerate, synth_full(), and synth_half().

Referenced by run_sync().

Here is the call graph for this function:



9.26.3.3 void mad_synth_init (struct mad_synth * *synth*)

Initializes synth struct.

Definition at line 38 of file synth.c.

References mad_pcm::channels, mad_pcm::length, mad_synth_mute(), mad_synth::pcm, mad_synth::phase, and mad_pcm::samplerate.

Referenced by run_sync().

Here is the call graph for this function:

9.26.3.4 void mad_synth_mute (struct mad_synth * *synth*)

Zeroes all polyphase filterbank values, resetting synthesis.

Definition at line 52 of file synth.c.

References mad_synth::filter, and s.

Referenced by mad_synth_init().

9.26.3.5 static void synth_full (struct mad_synth * *synth*, struct mad_frame const * *frame*, unsigned int *nch*, unsigned int *ns*) [static]

perform full frequency PCM synthesis.

Definition at line 563 of file synth.c.

References D, dct32(), mad_synth::filter, ML0, MLA, MLN, MLZ, mad_synth::pcm, mad_synth::phase, s, mad_pcm::samples, mad_frame::sbsample, and SHIFT.

Referenced by mad_synth_frame().

Here is the call graph for this function:

9.26.3.6 static void synth_half (struct mad_synth * *synth*, struct mad_frame const * *frame*, unsigned int *nch*, unsigned int *ns*) [static]

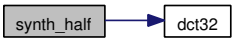
perform half frequency PCM synthesis.

Definition at line 699 of file synth.c.

References D, dct32(), mad_synth::filter, ML0, MLA, MLN, MLZ, mad_synth::pcm, mad_synth::phase, s, mad_pcm::samples, mad_frame::sbsample, and SHIFT.

Referenced by mad_synth_frame().

Here is the call graph for this function:



9.26.4 Variable Documentation

9.26.4.1 mad_fixed_t const D[17][32] [static]

Definition at line 551 of file synth.c.

Referenced by synth_full(), and synth_half().

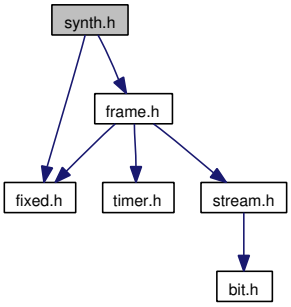
9.27 synth.h File Reference

Defines the mad_synth structure and methods (and its inner structure mad_pcm), to perform PCM synthesis.

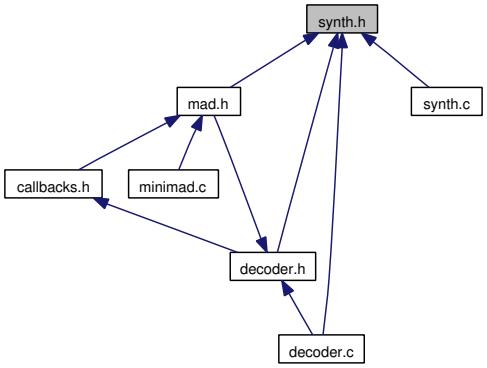
```
#include "fixed.h"
```

```
#include "frame.h"
```

Include dependency graph for synth.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct mad_pcm
- struct mad_synth

Defines

- #define mad_synth_finish(synth)

Enumerations

- enum { MAD_PCM_CHANNEL_SINGLE = 0 }
  single channel PCM selector
- enum { MAD_PCM_CHANNEL_DUAL_1 = 0, MAD_PCM_CHANNEL_DUAL_2 = 1 }
  dual channel PCM selector
- enum { MAD_PCM_CHANNEL_STEREO_LEFT = 0, MAD_PCM_CHANNEL_STEREO_RIGHT = 1 }
  stereo PCM selector

Functions

- void mad_synth_init (struct mad_synth *)
  Initializes synth struct.
- void mad_synth_mute (struct mad_synth *)
  Zeroes all polyphase filterbank values, resetting synthesis.
- void mad_synth_frame (struct mad_synth *, struct mad_frame const *)
  perform PCM synthesis of frame subband samples.

9.27.1 Detailed Description

Defines the mad_synth structure and methods (and its inner structure mad_pcm), to perform PCM synthesis.

Definition in file synth.h.

9.27.2 Define Documentation

9.27.2.1 #define mad_synth_finish(synth)

Definition at line 65 of file synth.h.

Referenced by run_sync().

9.27.3 Enumeration Type Documentation

9.27.3.1 anonymous enum

single channel PCM selector

Enumerator:

MAD_PCM_CHANNEL_SINGLE

Definition at line 47 of file synth.h.

9.27.3.2 anonymous enum

dual channel PCM selector

Enumerator:

MAD_PCM_CHANNEL_DUAL_1
MAD_PCM_CHANNEL_DUAL_2

Definition at line 52 of file synth.h.

9.27.3.3 anonymous enum

stereo PCM selector

Enumerator:

MAD_PCM_CHANNEL_STEREO_LEFT
MAD_PCM_CHANNEL_STEREO_RIGHT

Definition at line 58 of file synth.h.

9.27.4 Function Documentation

9.27.4.1 void mad_synth_frame (struct mad_synth *, struct mad_frame const *)

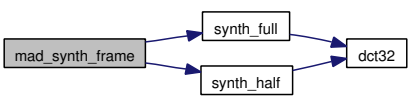
perform PCM synthesis of frame subband samples.

Definition at line 835 of file synth.c.

References mad_pcm::channels, mad_frame::header, mad_pcm::length, MAD_NCHANNELS, MAD_NSBSAMPLES, MAD_OPTION_HALFSAMPLERATE, mad_frame::options, mad_synth::pcm, mad_synth::phase, mad_header::samplerate, mad_pcm::samplerate, synth_full(), and synth_half().

Referenced by run_sync().

Here is the call graph for this function:



9.27.4.2 void mad_synth_init (struct mad_synth *)

Initializes synth struct.

Definition at line 38 of file synth.c.

References mad_pcm::channels, mad_pcm::length, mad_synth_mute(), mad_synth::pcm, mad_synth::phase, and mad_pcm::samplerate.

Referenced by run_sync().

Here is the call graph for this function:

9.27.4.3 void mad_synth_mute (struct mad_synth *)

Zeroes all polyphase filterbank values, resetting synthesis.

Definition at line 52 of file synth.c.

References mad_synth::filter, and s.

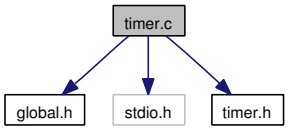
Referenced by mad_synth_init().

9.28 timer.c File Reference

Implements mad_timer_t methods, used when decoding header frames.

```
#include "global.h"
#include <stdio.h>
#include "timer.h"
```

Include dependency graph for timer.c:



Functions

- int mad_timer_compare (mad_timer_t timer1, mad_timer_t timer2)

  Indicates relative order of two timers.

- void mad_timer_negate (mad_timer_t *timer)

  Inverts the sign of a timer.

- mad_timer_t mad_timer_abs (mad_timer_t timer)

  Returns the absolute value of a timer.

- static void reduce_timer (mad_timer_t *timer)

  Carrys timer fraction into seconds.

- static unsigned long gcd (unsigned long num1, unsigned long num2)

  Computes greatest common denominator.

- static void reduce_rational (unsigned long *numer, unsigned long *denom)

  Converts rational expression to lowest terms.

- static unsigned long scale_rational (unsigned long numer, unsigned long denom, unsigned long scale)

  Solves numer/denom == ?/scale avoiding overflowing.

- void mad_timer_set (mad_timer_t *timer, unsigned long seconds, unsigned long numer, unsigned long denom)

  Sets timer to specific (positive) value.

- void mad_timer_add (mad_timer_t *timer, mad_timer_t incr)

  Adds one timer to another.

- void mad_timer_multiply (mad_timer_t *timer, signed long scalar)
  Multiplies a timer by a scalar value.
- signed long mad_timer_count (mad_timer_t timer, enum mad_units units)
  Returns timer value in selected units.
- unsigned long mad_timer_fraction (mad_timer_t timer, unsigned long denom)
  Returns fractional part of timer in arbitrary terms.
- void mad_timer_string (mad_timer_t timer, char *dest, char const *format, enum mad_units units, enum mad_units fracunits, unsigned long subparts)
  Writes a string representation of a timer using a template.

Variables

- mad_timer_t const mad_timer_zero = { 0, 0 }

9.28.1 Detailed Description

Implements mad_timer_t methods, used when decoding header frames.

Definition in file timer.c.

9.28.2 Function Documentation

9.28.2.1 static unsigned long gcd (unsigned long *num1*, unsigned long *num2*) [static]

Computes greatest common denominator.

Definition at line 101 of file timer.c.

Referenced by reduce_rational().

9.28.2.2 mad_timer_t mad_timer_abs (mad_timer_t *timer*)

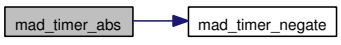
Returns the absolute value of a timer.

Definition at line 79 of file timer.c.

References mad_timer_negate(), and mad_timer_t::seconds.

Referenced by mad_timer_fraction(), and mad_timer_string().

Here is the call graph for this function:



9.28.2.3 void mad_timer_add (mad_timer_t * *timer*, mad_timer_t *incr*)

Adds one timer to another.

Definition at line 224 of file timer.c.

References mad_timer_t::fraction, MAD_TIMER_RESOLUTION, reduce_timer(), and mad_timer_t::seconds.

Referenced by mad_timer_multiply().

Here is the call graph for this function:

9.28.2.4 int mad_timer_compare (mad_timer_t *timer1*, mad_timer_t *timer2*)

Indicates relative order of two timers.

Definition at line 44 of file timer.c.

References mad_timer_t::fraction, and mad_timer_t::seconds.

9.28.2.5 signed long mad_timer_count (mad_timer_t *timer*, enum mad_units *units*)

Returns timer value in selected units.

Definition at line 262 of file timer.c.

References mad_timer_t::fraction, mad_timer_count(), MAD_TIMER_RESOLUTION, MAD_UNITS_11025_HZ, MAD_UNITS_12000_HZ, MAD_UNITS_16000_HZ, MAD_UNITS_22050_HZ, MAD_UNITS_23_976_FPS, MAD_UNITS_24000_HZ, MAD_UNITS_24_975_FPS, MAD_UNITS_24_FPS, MAD_UNITS_25_FPS, MAD_UNITS_29_97_FPS, MAD_UNITS_30_FPS, MAD_UNITS_32000_HZ, MAD_UNITS_44100_HZ, MAD_UNITS_47_952_FPS, MAD_UNITS_48000_HZ, MAD_UNITS_48_FPS, MAD_UNITS_49_95_FPS, MAD_UNITS_50_FPS, MAD_UNITS_59_94_FPS, MAD_UNITS_60_FPS, MAD_UNITS_75_FPS, MAD_UNITS_8000_HZ, MAD_UNITS_CENTISECONDS, MAD_UNITS_DECISECONDS, MAD_UNITS_HOURS, MAD_UNITS_MILLISECONDS, MAD_UNITS_MINUTES, MAD_UNITS_SECONDS, scale_rational(), and mad_timer_t::seconds.

Referenced by mad_timer_count(), and mad_timer_string().

Here is the call graph for this function:

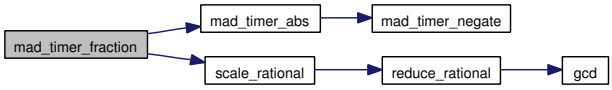
9.28.2.6 unsigned long mad_timer_fraction (mad_timer_t *timer*, unsigned long *denom*)

Returns fractional part of timer in arbitrary terms.

Definition at line 315 of file timer.c.

References mad_timer_t::fraction, mad_timer_abs(), MAD_TIMER_RESOLUTION, and scale_rational().

Here is the call graph for this function:



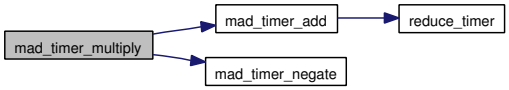
9.28.2.7 void mad_timer_multiply (mad_timer_t * *timer*, signed long *scalar*)

Multiplies a timer by a scalar value.

Definition at line 236 of file timer.c.

References mad_timer_add(), and mad_timer_negate().

Here is the call graph for this function:



9.28.2.8 void mad_timer_negate (mad_timer_t * *timer*)

Inverts the sign of a timer.

Definition at line 66 of file timer.c.

References mad_timer_t::fraction, MAD_TIMER_RESOLUTION, and mad_timer_t::seconds.

Referenced by mad_timer_abs(), and mad_timer_multiply().

9.28.2.9 void mad_timer_set (mad_timer_t * *timer*, unsigned long *seconds*, unsigned long *numer*, unsigned long *denom*)

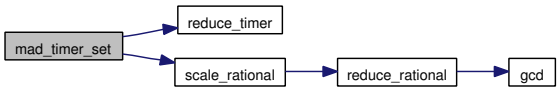
Sets timer to specific (positive) value.

Definition at line 153 of file timer.c.

References mad_timer_t::fraction, MAD_TIMER_RESOLUTION, reduce_timer(), scale_rational(), and mad_timer_t::seconds.

Referenced by mad_header_decode().

Here is the call graph for this function:



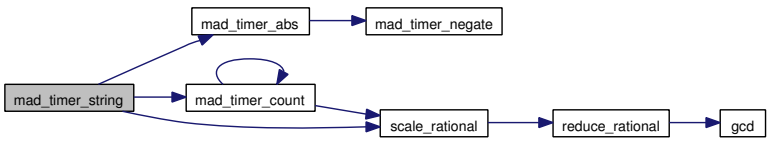
9.28.2.10 void mad_timer_string (mad_timer_t *timer*, char * *dest*, char const * *format*, enum mad_units *units*, enum mad_units *fracunits*, unsigned long *subparts*)

Writes a string representation of a timer using a template.

Definition at line 335 of file timer.c.

References mad_timer_t::fraction, m, mad_timer_abs(), mad_timer_count(), MAD_TIMER_RESOLUTION, MAD_UNITS_11025_HZ, MAD_UNITS_12000_HZ, MAD_UNITS_16000_HZ, MAD_UNITS_22050_HZ, MAD_UNITS_23_976_FPS, MAD_UNITS_24000_HZ, MAD_UNITS_24_975_FPS, MAD_UNITS_24_FPS, MAD_UNITS_25_FPS, MAD_UNITS_29_97_FPS, MAD_UNITS_30_FPS, MAD_UNITS_32000_HZ, MAD_UNITS_44100_HZ, MAD_UNITS_47_952_FPS, MAD_UNITS_48000_HZ, MAD_UNITS_48_FPS, MAD_UNITS_49_95_FPS, MAD_UNITS_50_FPS, MAD_UNITS_59_94_FPS, MAD_UNITS_60_FPS, MAD_UNITS_75_FPS, MAD_UNITS_8000_HZ, MAD_UNITS_CENTISECONDS, MAD_UNITS_DECISECONDS, MAD_UNITS_HOURS, MAD_UNITS_MILLISECONDS, MAD_UNITS_MINUTES, MAD_UNITS_SECONDS, scale_rational(), and mad_timer_t::seconds.

Here is the call graph for this function:



9.28.2.11 static void reduce_rational (unsigned long * *numer*, unsigned long * *denom*) [static]

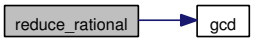
Converts rational expression to lowest terms.

Definition at line 118 of file timer.c.

References assert, and gcd().

Referenced by scale_rational().

Here is the call graph for this function:



9.28.2.12 static void reduce_timer (mad_timer_t * *timer*) [static]

Carrys timer fraction into seconds.

Definition at line 91 of file timer.c.

References mad_timer_t::fraction, MAD_TIMER_RESOLUTION, and mad_timer_t::seconds.

Referenced by mad_timer_add(), and mad_timer_set().

9.28.2.13 static unsigned long scale_rational (unsigned long *numer*, unsigned long *denom*, unsigned long *scale*) [static]

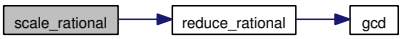
Solves numer/denom == ?/scale avoiding overflowing.

Definition at line 134 of file timer.c.

References assert, and reduce_rational().

Referenced by mad_timer_count(), mad_timer_fraction(), mad_timer_set(), and mad_timer_string().

Here is the call graph for this function:



9.28.3 Variable Documentation

9.28.3.1 mad_timer_t const mad_timer_zero = { 0, 0 }

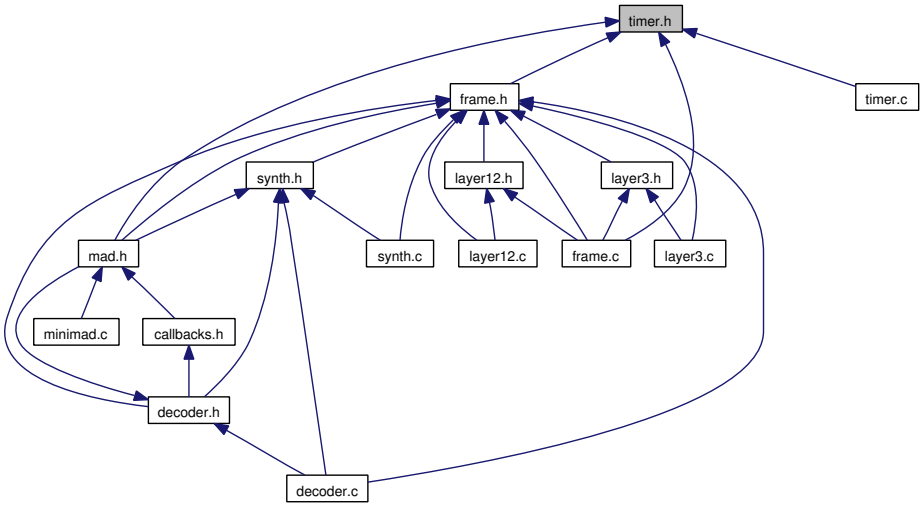
Definition at line 39 of file timer.c.

Referenced by mad_header_init().

9.29 timer.h File Reference

Declares mad_timer_t structure and methods, used when decoding header frames.

This graph shows which files directly or indirectly include this file:



Data Structures

- struct mad_timer_t

Defines

- #define MAD_TIMER_RESOLUTION 352800000UL
- #define mad_timer_reset(timer) ((void) (*(timer) = mad_timer_zero))
- #define mad_timer_sign(timer) mad_timer_compare((timer), mad_timer_zero)

Enumerations

- enum mad_units {

  MAD_UNITS_HOURS = -2, MAD_UNITS_MINUTES = -1, MAD_UNITS_-SECONDS = 0, MAD_UNITS_DECISECONDS = 10,

  MAD_UNITS_CENTISECONDS = 100, MAD_UNITS_MILLISECONDS = 1000, MAD_UNITS_8000_HZ = 8000, MAD_UNITS_11025_HZ = 11025,

  MAD_UNITS_12000_HZ = 12000, MAD_UNITS_16000_HZ = 16000, MAD_UNITS_22050_HZ = 22050, MAD_UNITS_24000_HZ = 24000,

  MAD_UNITS_32000_HZ = 32000, MAD_UNITS_44100_HZ = 44100, MAD_UNITS_48000_HZ = 48000, MAD_UNITS_24_FPS = 24,

MAD_UNITS_25_FPS = 25, MAD_UNITS_30_FPS = 30, MAD_UNITS_48_-FPS = 48, MAD_UNITS_50_FPS = 50,

MAD_UNITS_60_FPS = 60, MAD_UNITS_75_FPS = 75, MAD_UNITS_23_-976_FPS = -24, MAD_UNITS_24_975_FPS = -25,

MAD_UNITS_29_97_FPS = -30, MAD_UNITS_47_952_FPS = -48, MAD_-UNITS_49_95_FPS = -50, MAD_UNITS_59_94_FPS = -60 }

Functions

- int mad_timer_compare (mad_timer_t, mad_timer_t)

  Indicates relative order of two timers.

- void mad_timer_negate (mad_timer_t *)

  Inverts the sign of a timer.

- mad_timer_t mad_timer_abs (mad_timer_t)

  Returns the absolute value of a timer.

- void mad_timer_set (mad_timer_t *, unsigned long, unsigned long, unsigned long)

  Sets timer to specific (positive) value.

- void mad_timer_add (mad_timer_t *, mad_timer_t)

  Adds one timer to another.

- void mad_timer_multiply (mad_timer_t *, signed long)

  Multiplies a timer by a scalar value.

- signed long mad_timer_count (mad_timer_t, enum mad_units)

  Returns timer value in selected units.

- unsigned long mad_timer_fraction (mad_timer_t, unsigned long)

  Returns fractional part of timer in arbitrary terms.

- void mad_timer_string (mad_timer_t, char *, char const *, enum mad_units, enum mad_units, unsigned long)

  Writes a string representation of a timer using a template.

Variables

- mad_timer_t const mad_timer_zero

9.29.1 Detailed Description

Declares mad_timer_t structure and methods, used when decoding header frames.

Definition in file timer.h.

9.29.2 Define Documentation

9.29.2.1 #define mad_timer_reset(timer) ((void) (*(timer) = mad_timer_zero))

Definition at line 84 of file timer.h.

9.29.2.2 #define MAD_TIMER_RESOLUTION 352800000UL

Definition at line 34 of file timer.h.

Referenced by mad_timer_add(), mad_timer_count(), mad_timer_fraction(), mad_timer_negate(), mad_timer_set(), mad_timer_string(), and reduce_timer().

9.29.2.3 #define mad_timer_sign(timer) mad_timer_compare((timer), mad_timer_zero)

Definition at line 88 of file timer.h.

9.29.3 Enumeration Type Documentation

9.29.3.1 enum mad_units

Enumerator:

MAD_UNITS_HOURS

MAD_UNITS_MINUTES

MAD_UNITS_SECONDS

MAD_UNITS_DECISECONDS

MAD_UNITS_CENTISECONDS

MAD_UNITS_MILLISECONDS

MAD_UNITS_8000_HZ

MAD_UNITS_11025_HZ

MAD_UNITS_12000_HZ

MAD_UNITS_16000_HZ

MAD_UNITS_22050_HZ

MAD_UNITS_24000_HZ

MAD_UNITS_32000_HZ

MAD_UNITS_44100_HZ

MAD_UNITS_48000_HZ

MAD_UNITS_24_FPS

MAD_UNITS_25_FPS

MAD_UNITS_30_FPS

MAD_UNITS_48_FPS

MAD_UNITS_50_FPS

MAD_UNITS_60_FPS

MAD_UNITS_75_FPS

MAD_UNITS_23_976_FPS

MAD_UNITS_24_975_FPS

MAD_UNITS_29_97_FPS

MAD_UNITS_47_952_FPS

MAD_UNITS_49_95_FPS

MAD_UNITS_59_94_FPS

Definition at line 36 of file timer.h.

9.29.4 Function Documentation

9.29.4.1 mad_timer_t mad_timer_abs (mad_timer_t)

Returns the absolute value of a timer.

Definition at line 79 of file timer.c.

References mad_timer_negate(), and mad_timer_t::seconds.

Referenced by mad_timer_fraction(), and mad_timer_string().

Here is the call graph for this function:

9.29.4.2 void mad_timer_add (mad_timer_t *, mad_timer_t)

Adds one timer to another.

Definition at line 224 of file timer.c.

References mad_timer_t::fraction, MAD_TIMER_RESOLUTION, reduce_timer(), and mad_timer_t::seconds.

Referenced by mad_timer_multiply().

Here is the call graph for this function:

9.29.4.3 int mad_timer_compare (mad_timer_t, mad_timer_t)

Indicates relative order of two timers.

Definition at line 44 of file timer.c.

References mad_timer_t::fraction, and mad_timer_t::seconds.

9.29.4.4 signed long mad_timer_count (mad_timer_t, enum *mad_units*)

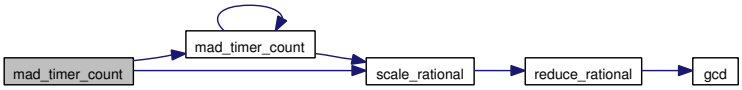
Returns timer value in selected units.

Definition at line 262 of file timer.c.

References mad_timer_t::fraction, mad_timer_count(), MAD_TIMER_RESOLUTION, MAD_UNITS_11025_HZ, MAD_UNITS_12000_HZ, MAD_UNITS_16000_HZ, MAD_UNITS_22050_HZ, MAD_UNITS_23_976_FPS, MAD_UNITS_24000_HZ, MAD_UNITS_24_975_FPS, MAD_UNITS_24_FPS, MAD_UNITS_25_FPS, MAD_UNITS_29_97_FPS, MAD_UNITS_30_FPS, MAD_UNITS_32000_HZ, MAD_UNITS_44100_HZ, MAD_UNITS_47_952_FPS, MAD_UNITS_48000_HZ, MAD_UNITS_48_FPS, MAD_UNITS_49_95_FPS, MAD_UNITS_50_FPS, MAD_UNITS_59_94_FPS, MAD_UNITS_60_FPS, MAD_UNITS_75_FPS, MAD_UNITS_8000_HZ, MAD_UNITS_CENTISECONDS, MAD_UNITS_DECISECONDS, MAD_UNITS_HOURS, MAD_UNITS_MILLISECONDS, MAD_UNITS_MINUTES, MAD_UNITS_SECONDS, scale_rational(), and mad_timer_t::seconds.

Referenced by mad_timer_count(), and mad_timer_string().

Here is the call graph for this function:



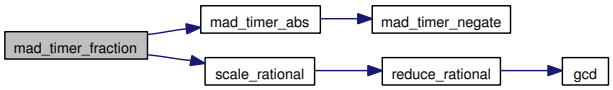
9.29.4.5 unsigned long mad_timer_fraction (mad_timer_t, unsigned *long*)

Returns fractional part of timer in arbitrary terms.

Definition at line 315 of file timer.c.

References mad_timer_t::fraction, mad_timer_abs(), MAD_TIMER_RESOLUTION, and scale_rational().

Here is the call graph for this function:



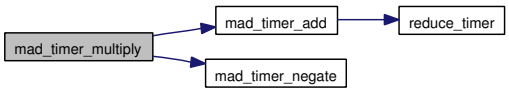
9.29.4.6 void mad_timer_multiply (mad_timer_t *, signed *long*)

Multiplies a timer by a scalar value.

Definition at line 236 of file timer.c.

References mad_timer_add(), and mad_timer_negate().

Here is the call graph for this function:



9.29.4.7 void mad_timer_negate (mad_timer_t *)

Inverts the sign of a timer.

Definition at line 66 of file timer.c.

References mad_timer_t::fraction, MAD_TIMER_RESOLUTION, and mad_timer_t::seconds.

Referenced by mad_timer_abs(), and mad_timer_multiply().

9.29.4.8 void mad_timer_set (mad_timer_t *, unsigned *long*, unsigned *long*, unsigned *long*)

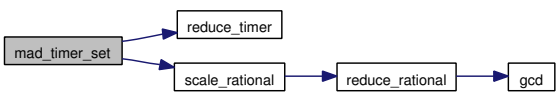
Sets timer to specific (positive) value.

Definition at line 153 of file timer.c.

References mad_timer_t::fraction, MAD_TIMER_RESOLUTION, reduce_timer(), scale_rational(), and mad_timer_t::seconds.

Referenced by mad_header_decode().

Here is the call graph for this function:



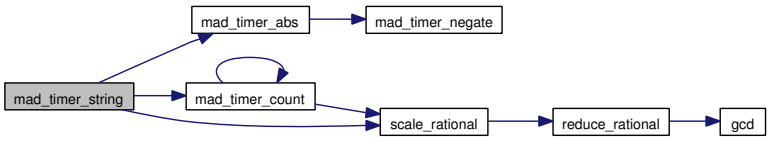
9.29.4.9 void mad_timer_string (mad_timer_t, char *, char const *, enum *mad_units*, enum *mad_units*, unsigned *long*)

Writes a string representation of a timer using a template.

Definition at line 335 of file timer.c.

References mad_timer_t::fraction, m, mad_timer_abs(), mad_timer_count(), MAD_TIMER_RESOLUTION, MAD_UNITS_11025_HZ, MAD_UNITS_12000_HZ, MAD_UNITS_16000_HZ, MAD_UNITS_22050_HZ, MAD_UNITS_23_976_FPS, MAD_UNITS_24000_HZ, MAD_UNITS_24_975_FPS, MAD_UNITS_24_FPS, MAD_UNITS_25_FPS, MAD_UNITS_29_97_FPS, MAD_UNITS_30_FPS, MAD_UNITS_32000_HZ, MAD_UNITS_44100_HZ, MAD_UNITS_47_952_FPS, MAD_UNITS_48000_HZ, MAD_UNITS_48_FPS, MAD_UNITS_49_95_FPS, MAD_UNITS_50_FPS, MAD_UNITS_59_94_FPS, MAD_UNITS_60_FPS, MAD_UNITS_75_FPS, MAD_UNITS_8000_HZ, MAD_UNITS_CENTISECONDS, MAD_UNITS_DECISECONDS, MAD_UNITS_HOURS, MAD_UNITS_MILLISECONDS, MAD_UNITS_MINUTES, MAD_UNITS_SECONDS, scale_rational(), and mad_timer_t::seconds.

Here is the call graph for this function:



9.29.5 Variable Documentation

9.29.5.1 mad_timer_t const mad_timer_zero

Definition at line 39 of file timer.c.

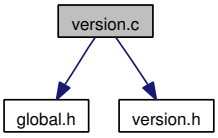
Referenced by mad_header_init().

9.30 version.c File Reference

Initializes some libmad version-related constants.

```
#include "global.h"
#include "version.h"
```

Include dependency graph for version.c:



Variables

- char const mad_version [ ] = "MPEG Audio Decoder " MAD_VERSION
- char const mad_copyright [ ] = "Copyright (C) " MAD_PUBLISHYEAR " " MAD_AUTHOR
- char const mad_author [ ] = MAD_AUTHOR " <" MAD_EMAIL ">"
- char const mad_build [ ] = ""

9.30.1 Detailed Description

Initializes some libmad version-related constants.

Definition in file version.c.

9.30.2 Variable Documentation

9.30.2.1 char const mad_author[ ] = MAD_AUTHOR " <" MAD_EMAIL ">"

Definition at line 35 of file version.c.

9.30.2.2 char const mad_build[ ] = ""

Definition at line 37 of file version.c.

9.30.2.3 char const mad_copyright[ ] = "Copyright (C) " MAD_PUBLISHYEAR " " MAD_AUTHOR

Definition at line 34 of file version.c.

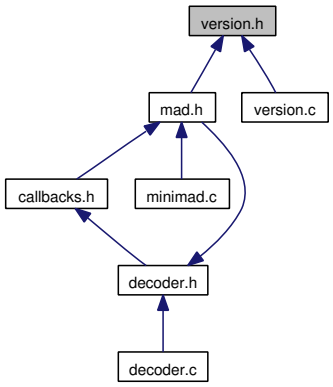
9.30.2.4 char const mad_version[ ] = "MPEG Audio Decoder " MAD_VERSION

Definition at line 33 of file version.c.

9.31 version.h File Reference

Defines various libmad version-related constants.

This graph shows which files directly or indirectly include this file:



Defines

- #define MAD_VERSION_MAJOR 0
- #define MAD_VERSION_MINOR 15
- #define MAD_VERSION_PATCH 1
- #define MAD_VERSION_EXTRA " (beta)"
- #define MAD_VERSION_STRINGIZE(str) #str
- #define MAD_VERSION_STRING(num) MAD_VERSION_STRINGIZE(num)
- #define MAD_VERSION
- #define MAD_PUBLISHYEAR "2000-2004"
- #define MAD_AUTHOR "Underbit Technologies, Inc."
- #define MAD_EMAIL "info at underbit com"

Variables

- char const mad_version [ ]
- char const mad_copyright [ ]
- char const mad_author [ ]
- char const mad_build [ ]

9.31.1 Detailed Description

Defines various libmad version-related constants.

Definition in file version.h.

9.31.2 Define Documentation

9.31.2.1 #define MAD_AUTHOR "Underbit Technologies, Inc."

Definition at line 42 of file version.h.

9.31.2.2 #define MAD_EMAIL "info at underbit com"

Definition at line 43 of file version.h.

9.31.2.3 #define MAD_PUBLISHYEAR "2000-2004"

Definition at line 41 of file version.h.

9.31.2.4 #define MAD_VERSION

Value:

```
MAD_VERSION_STRING(MAD_VERSION_MAJOR) "."  \
                                  MAD_VERSION_STRING(MAD_VERSION_MINOR) "."  \
                                  MAD_VERSION_STRING(MAD_VERSION_PATCH)  \
                                  MAD_VERSION_EXTRA
```

Definition at line 36 of file version.h.

9.31.2.5 #define MAD_VERSION_EXTRA " (beta)"

Definition at line 31 of file version.h.

9.31.2.6 #define MAD_VERSION_MAJOR 0

Definition at line 28 of file version.h.

9.31.2.7 #define MAD_VERSION_MINOR 15

Definition at line 29 of file version.h.

9.31.2.8 #define MAD_VERSION_PATCH 1

Definition at line 30 of file version.h.

9.31.2.9 #define MAD_VERSION_STRING(num) MAD_VERSION_STRINGIZE(num)

Definition at line 34 of file version.h.

9.31.2.10 #define MAD_VERSION_STRINGIZE(str) #str

Definition at line 33 of file version.h.

9.31.3 Variable Documentation

9.31.3.1 char const mad_author[ ]

Definition at line 35 of file version.c.

9.31.3.2 char const mad_build[ ]

Definition at line 37 of file version.c.

9.31.3.3 char const mad_copyright[ ]

Definition at line 34 of file version.c.

9.31.3.4 char const mad_version[ ]

Definition at line 33 of file version.c.