LAME Technical FAQ




==========

1.  Why is a decoded MP3 longer than the original .wav file?

Because LAME (and all other MDCT based encoders) add padding to the beginning and end of each song.  For an explination of why, see the questions below.

LAME embeds the amount of padding in the ancillary data of the first frame of the MP3 file. (LAME INFO tag). The LAME decoder will use this information to remove the leading padding of an MP3 file.

Modifications to the decoder so that it will also remove the trailing padding have not yet been made.

==========

2.  Why does LAME add silence to the beginning each song?

This is because of several factors:

DECODER DELAY AT START OF FILE:

All *decoders* I have tested introduce a delay of 528 samples.  That is, after decoding an mp3 file, the output will have 528 samples of 0's appended to the front.  This is because the standard MDCT/filterbank routines used by the ISO have a 528 sample delay.  It would be possible to write a MDCT/filterbank routine with a 0 sample delay (see description of Takehiro's MDCT/filterbank routine used in LAME encoding below) but I dont know that anyone has done this. Furthermore, because of the overlapped nature of MDCT frames, the first half of the first granule (1 granule=576 samples) doesn't have a previous frame to overlap with, resulting in attenuation of the first N samples.  The value of N depends on the window type.  For "STOP_TYPE" and "SHORT_TYPE", N=96, while for "START_TYPE" and "NORMAL_TYPE", N=288.  The first frame produced by LAME 3.56 and up will always be of STOP_TYPE or SHORT_TYPE.

ENCODER DELAY AT START OF FILE:

ISO based encoders (BladeEnc, 8hz-mp3, etc) use a MDCT/filterbank routine similar to the one used in decoding, and thus also introduce their own 528 sample delay.  A .wav file encoded & decoded will have a

1056 sample delay (1056 samples will be appended to the beginning).

(actually, the observed delay is 1057 samples.  For even more
technical discussions about this, see the mp3encoder mailing list
archive)

The FhG encoder (at highest quality) introduces a 1160 sample delay,
for a total encoding/decoding delay of 1688 samples.  I haven't tested
Xing.

Starting with LAME 3.55, we have a new MDCT/filterbank routine written
by Takehiro Tominaga with a 48 sample delay.  With even more rewriting,
this could be reduced to 0.  And there is no reason an inverse routine
could not be used in a decoder.  However, there are a few problems
with using such a short delay:

1.)  The 96 samples of the first frame are attenuated by the MDCT
     window.  If the encoder delay is greater than 96, this window will
     have no effect since the first 96 samples are all padding.  With a
     48 sample encoder delay, the first 48 samples will be improperly
     attenuated. (.001 seconds worth of data at 44.1kHz).

2.)  Filterbank contamination problems.  This it rather technical, see
     below.

3.)  In LAME, psycho-acoustics for the first 576 granule are not correct.
     This could be fixed, but at the expense of adding more buffering
     and code complexity.

If points 1. 2. or 3. do not bother you, you can decrease the
the encoder delay by setting ENCDELAY in encoder.h.  The default
right now is 576.

More technical details on very short ENCDELAY:
Here is an example:

```
                    granule 1         granule 2
                    576 samples       576 samples
                | 192 | 192 | 192 | 192 | 192 | 192 |
data:           <    all zeros'   >< real  data     >
short block   <---------->
                    <--------->
                           <--------->
end block                       <---------------------->

output:                           | 192 | 192 | 192 |
```

granule 2 is the first granule that is encoded.  granule 1 is
the fictitious previous granule that doesn't really exist, except
that the output of the decoder for granule 2 is going to be combined with
the data in the buffer which would have held the decoded output of
granule 1.  I will assume the decoder initializes this buffer with
zeros.

Lets ignore quantization.  The lapped MDCT followed
by the IMDCT is lossless.  That means that the IMDCT output

```
from granule 2  when added to the IMDCT output from granule 1
is identical to the input.

In our case, the decoder just sets the granule 1 IMDCT output to all
0's because it never actually computes this and is just initializing a
buffer.  But the output of granule 2 IMDCT is computed correctly.

output = granule_1_output + granule_2_output

granule_1_output:  encoder uses all 0's, which is incorrect since
                   the MDCT (if it was performed) would have seen
                   some of the data in granule 2.

granule_2_output:  correct

Therefor, the output will be correct *except* where it uses data from
granule 1, but this can effect at most the first 96 samples.

However, the polyphase filterbank is another story:

The data (with first 96 samples corrupt) is then sent to the
inverse polyphase filterbank.  I dont know much about how this
albatross works, but I think it has an effective window length of
512.  So the bad data in the first 96 samples can corrupt
samples up to 96+512.




=============================================================================
3.  Why does LAME add silence to the end of each song?

Extra padding at the end of a file can be caused by a couple of things:

1.  Because the MDCT's are overlapped, it looks something like this:

<--576 MDCT coefficients--><--576 MDCT coefficients--><--576 MDCT coefficients-->
            <-- 576 samples PCM output --><-- 576 samples PCM output -->

   So no matter where you truncate your MP3 file, the last 288 samples of
   that granule will not be decoded.  So LAME appends 288 samples of
   padding to the input file to guarantee all input samples will be
   decoded.


2. If the number of samples is not an exact multiple of 1152,
   then last frame of data is padded with 0's so that it has 1152 samples.


Before lame3.56, we just added a few extra frames to make sure all
internal buffers would be flushed.  In lame3.56, we tried to pad
with the exact minimum number of samples needed.  And in lame3.80,
we finally fixed the bitstream flushing so that the final mp3
frame is properly padded with ancillary data.


=============================================================================
4.  Why cant MP3 files be seamlessly spliced together?
```

There are several reasons this is *very* difficult (but not impossible):

The MP3 data for frame N is not stored in frame N, but can be spread
over several frames.  In a typical case, the data for frame N will
have 20% of it stored in frame N-1 and 80% stored in frame N.
If the encoder builds up a large bit reservoir,
the data for frame N can actually be stored 4088 bits back in
the bitstream.  Then if a very hard-to-encode passage comes up,
then the encoder is free to use the normal bits for this frame
plus up to 4088 more.  The resulting data will then take up
several frames.   The starting negative offset
in the bitstream for the data associated with a given frame in bytes is
given by main_data_begin.

Thus chopping a mp3 file on a frame boundary will almost always result
in the corruption of the data in that frame.   mpg123 will report
such errors as "cant seek past beginning of file" or something
like that.

A proper cut-and-past job could be done, but it would have to
separate all the data from the frame headers, and then
replace the frame headers in the correct location in the new
stream.  One problem:  this may generate data for frame N that
is stored too far back, say 4100 bits back.  In that case, the
main_data_begin field will be incorrect since it can be at most 4088.

Two possible solutions:

1. Create mp3's with the --nores option in LAME,
(disabling the bit reservoir and reducing quality somewhat),
these mp3 files can be simply cut and pasted on frame boundaries.

2. Use VBR and overlapping encodes.  For example:
   stream A = encode frames 0-99
   stream B = encode frames 97-200

   First remove the frames 97,98 and 99 from stream B.  It is
   important to use overlapping encoding because of the
   psycho-acoustics.  Then take frame 100 in stream B.  Most of the
   time, some data for frame 100 will be stored in frame 99.  Take a
   look at frame 99 from stream A.  If there is enough space, take the
   frame100 data which was stored in stream B/frame 99, and store it
   in stream A/frame 99.  If there is not enough space, replace frame
   100 with a higher bitrate frame to make enough space.

   Now stream A and stream B can be concatenated together.


Note that MP3 stores MDCT coefficients which represent 1152 samples,
but they are overlapped by 50%.  So for example:

```
frame N     <  0...1152    >
frame N+1         <  576...1727   >
frame N+2                 <   1152...2304   >
```

You need to add all the data together to complete the frame.  The
complete frame of samples 576-1727 needs frame N, N+1 and N+2.

```
==========================================================================
5. What is the size of a MPEG1/2 frame?

The number of bits/frame is:  frame_size*bit_rate/sample_rate.
For MPEG1, frame_size = 1152 samples/frame
For MPEG2, frame_size =  576 samples/frame

For example,

320k bits/second) *  ( 1152 samples/frame )  /  (48k samples/second) =

1152*320/48  bits/frame

largest mpeg1 frame:  1152*320/32 = 1440 bytes/frame
largest mpeg2 frame:   576*160/16 =  720 bytes/frame

For some sample rates the bits/frame will not come out as a integer
number of bytes.  In those cases round down to the nearest byte,
unless the padding bit is set in which case round up.  (padding makes
the frame one byte larger) The encoder is supposed to use padding in
selected frames to keep the average bitrate as close as possible to
the specified bitrate.

==========================================================================
6. Does LAME use any MP3 patented technology?

LAME, as the name says, is *not* an encoder.  LAME is a development
project which uses the open source model to improve MP3 technology.
Many people believe that compiling this code and distributing an
encoder which uses this code would violate some patents (in the US,
Europe and Japan).  However, *only* a patent lawyer is qualified to
make this determination.  The LAME project tries to avoid all these
legal issues by only releasing source code, much like the ISO
distributes MP3 "demonstration" source code.  Source code is
considered as speech, which may contain descriptions of patented
technology.  Descriptions of patents are in the public
domain.

Several companies plan on releasing encoders based on LAME, and
they intend to obtain all the appropriate patent licenses.  At least
one company is now shipping a fully licensed version of LAME with
their portable MP3 player.

Note that under German Patent Law, §11(1) a patent doesn't cover
private acts with non-industrial purposes. Probably interesting for
developers is that a patent doesn't cover acts with experimental
purposes, that aim at the object of the patented invention (§11(2)).
```