Logo

Description automatically generated

Ittiam Systems (P) Ltd,

The Consulate, 1 Richmond Road,

Bangalore 560 025, India

**ITTIAM-FFMPEG-MPEGH-ENC-UG**

User Guide

|  |  |
| --- | --- |
| Document Number | ITTIAM-FFMPEG-MPEGH-ENC-UG |
| Version | V1.0 |
| Date | February 17, 2023 |
|  | |

Notice

This document contains information, which is the proprietary property of Ittiam Systems. This document is received in confidence and its contents cannot be disclosed or copied without the prior written consent of Ittiam Systems. Ittiam Systems retains the right to make changes to this document at any time, without notice. Ittiam Systems makes no warranty for the use of this document.

Ittiam Systems reserves the right to make changes to its products or discontinue any of its products or offerings without notice.

Ittiam warrants the performance of its products to the specifications applicable at the time of sale in accordance with Ittiam’s standard warranty.

Disclaimer

In addition to the warranty disclaimers contained in these licenses, Ittiam Systems Pvt. Ltd. makes the following disclaimers regarding the third-party components on behalf of itself, its affiliates and investors, its contributors, the copyright holders, and the licensors of the third-party components:

YOUR USE OF THE SOFTWARE MAY REQUIRE ADDITIONAL PATENT LICENSE(S) BY THIRD

PARTIES, INCLUDING, WITHOUT LIMITATION, THE PATENT OWNERS LISTED AT

<https://www.via-corp.com/licensing/mpeg-h/mpeg-h-3d-licensors/> TO THE

FULLEST EXTENT PERMITTED UNDER APPLICABLE LAW, THE SOFTWARE IS PROVIDED

“AS IS,” AND ANY REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER ORAL

OR WRITTEN, WHETHER EXPRESS, IMPLIED, OR ARISING BY STATUTE, CUSTOM, COURSE

OF DEALING, OR TRADE USAGE, INCLUDING WITHOUT LIMITATION THE IMPLIED

WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE,

AND NONINFRINGEMENT, ARE DISCLAIMED. IN NO EVENT WILL ITTIAM (AND ITS

AFFILIATES AND INVESTORS), THE COPYRIGHT OWNERS, CONTRIBUTORS, OR LICENSORS,

BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR

CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF

SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS

INTERRUPTION), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN

CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

ARISING IN ANY WAY OUT OF THE USE OF THE THIRD-PARTY COMPONENTS, EVEN IF

ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Revision History

|  |  |
| --- | --- |
| Date | Changes |
| February 17, 2023 | Original |

Copyright © 2023, Ittiam Systems (P) Ltd

Contents

[1. Introduction 5](#_Toc127441786)

[2. Integration of Plugin for Linux or MAC OS 6](#_Toc127441787)

[2.1 Prerequisite 6](#_Toc127441788)

[2.2 Creating Library Shared Object 6](#_Toc127441789)

[2.3 Plugin Integration to FFMPEG 6](#_Toc127441790)

[2.3.1 Integration Using a Patch File 6](#_Toc127441791)

[2.3.1.1 Applying the Patch 6](#_Toc127441792)

[2.3.2 Manual Integration 7](#_Toc127441793)

[2.3.2.1 Steps for Integrating MPEGH Encoder Plugin into FFMPEG Framework Manually 7](#_Toc127441794)

[2.3.2.2 Steps for Adding MP4 Support 8](#_Toc127441795)

[2.4 Steps for Configure, Build and Execute 11](#_Toc127441796)

[3. Integration of Plugin for Windows 12](#_Toc127441797)

[3.1 Prerequisite 12](#_Toc127441798)

[3.2 Creating DLL 12](#_Toc127441799)

[3.3 Plugin Integration to FFMPEG 12](#_Toc127441800)

[3.3.1 Integration Using a Patch File 12](#_Toc127441801)

[3.3.1.1 Applying the Patch 12](#_Toc127441802)

[3.3.2 Manual Integration 13](#_Toc127441803)

[3.3.2.1 Steps for Integrating MPEGH Encoder Plugin into FFMPEG Framework Manually 13](#_Toc127441804)

[3.3.2.2 Steps for Adding MP4 Support 14](#_Toc127441805)

[3.4 Steps for Configure, Build and Execute 17](#_Toc127441806)

[4. Usage of Plugin 18](#_Toc127441807)

[4.1 Command Line Arguments 18](#_Toc127441808)

[4.2 Command Usage 18](#_Toc127441809)

[4.2.1 For MHAS Output Format 18](#_Toc127441810)

[4.2.2 For MP4 Output Format 19](#_Toc127441811)

[5. Notes and Limitations 21](#_Toc127441812)

# Introduction

FFMPEG plugins provide an efficient and fast way for media application development. Ittiam FFMPEG plugin benefit the developers for application development using the standard FFMPEG APIs.

This document describes steps for integration of MPEGH Encoder plugin code provided by Ittiam with FFMPEG – 5.1. The FFMPEG source code is available at <https://git.ffmpeg.org/ffmpeg.git>.

# Integration of Plugin for Linux or MAC OS

## Prerequisite

Git needs to be installed in host PC. It can be downloaded from the below link:

<https://git-scm.com/downloads>.

## Creating Library Shared Object

FFMPEG requires encoder library in shared object format (.so format). Shared object can be created using the below steps:

* Download libmpeghe source code from <https://github.com/ittiam-systems/libmpeghe>.
* Copy the patch file libmpeghe/ffmpeg\_plugin/linux\_mpegh\_enc\_lib.patch to libmpeghe directory.
* Run the below command in git bash from libmpeghe directory to apply patch:

git apply linux\_mpegh\_enc\_lib.patch

* With patch applied, libmpeghe/build/makefile/Makefile\_lib file gets updated to create encoder library as shared object.
* Build the library using the below command from libmpeghe/build/makefile/ directory:

make -f Makefile\_lib clean all

* The shared object libia\_mpeghe.so is generated in the libmpeghe/build/makefile/ directory.

## Plugin Integration to FFMPEG

The MPEG-H encoder plugin can be integrated into FFMPEG in two methods:

* Using the patch file
* Manual integration.

### Integration Using a Patch File

#### Applying the Patch

* Follow the steps mentioned in [section 2.2](#_Creating_Library_Shared) if not done yet.
* Download the FFMPEG source code version 5.1 from <https://git.ffmpeg.org/ffmpeg.git> using the below commands in Git bash.

git clone <https://git.ffmpeg.org/ffmpeg.git>

cd ffmpeg

git checkout release/5.1

* Copy the patch file libmpeghe/ffmpeg\_plugin/ffmpeg\_mpegh\_enc.patch to ffmpeg directory.
* To apply the patch ffmpeg\_mpegh\_enc, use the command given below:

git apply ffmpeg\_mpegh\_enc.patch

* As the shared object libia\_mpeghe.so is not part of the patch, it needs to be copied to ffmpeg folder manually.

Copy libia\_mpeghe.sofrom libmpeghe/build/makefile/directory toffmpeg/.

* Move to [Steps for Configure, Build and Execute](#_Steps_for_Configure,).

### Manual Integration

#### Steps for Integrating MPEGH Encoder Plugin into FFMPEG Framework Manually

1. Follow the steps mentioned in section 2.2 if not done yet.
2. Download the FFMPEG source code version 5.1 from <https://git.ffmpeg.org/ffmpeg.git> using the below commands in Git bash.

git clone <https://git.ffmpeg.org/ffmpeg.git>

cd ffmpeg

git checkout release/5.1

1. Copy *libia\_mpeghe.so* from libmpeghe/build/makefile/directory to *ffmpeg/*.
2. Copy all *C source* filesfrom libmpeghe*/*ffmpeg\_plugin*/ffmpeg/src/* to *ffmpeg/libavcodec* folder.
3. Copy all *header* files from libmpeghe*/*ffmpeg\_plugin*/ffmpeg/include/* to *ffmpeg/libavcodec* folder.
4. Search for EXTRALIBS tag in configuration file (File name: *configure*) and pass the argument ‘-lia\_mpeghe -L.’. The command should look like as below.

EXTRALIBS=$extralibs **–lia\_mpeghe –L.**

This is for linking Ittiam MPEGH Encoder library.

1. Add the below line in file *ffmpeg/libavcodec/Makefile* under section ‘# decoders/ encoders’:

**OBJS-$(CONFIG\_IA\_MPEGH\_ENCODER) += ia\_mpegh\_enc.o impeghe\_error.o**

1. Add the below line of code in the file *libavcodec/allcodecs.c* under section ‘*/\* external libraries \*/*’:

**extern const FFCodec ff\_ia\_mpegh\_encoder;**

Ittiam recommends referring https://trac.ffmpeg.org/wiki/CompilationGuide for further information on compilation and setup.

#### Steps for Adding MP4 Support

Below are the steps to add MPEGH format support to the existing FFMPEG’s MP4 multiplexer.

1. In *libavformat/movenc.c* file:

**Update #1**

**Line:1162 (**Add given function before static int mov\_write\_btrt\_tag(AVIOContext \*pb, MOVTrack \*track)**)**

**static int mov\_write\_mhac\_tag(AVIOContext \*pb, MOVTrack \*track)**

**{**

**// Size of mhaC**

**avio\_wb32(pb,track->par->extradata\_size + 8 -1);**

**ffio\_wfourcc(pb, "mhaC");**

**// Writing the buffer**

**avio\_write(pb,track->par->extradata,track->par->extradata\_size - 1);**

**return 1;**

**}**

**Update #2**

**Function:** static int mov\_write\_audio\_tag(AVFormatContext \*s, AVIOContext \*pb, MOVMuxContext \*mov, MOVTrack \*track)

**Line: 1293**

if (track->mode == MODE\_MOV &&

(track->par->codec\_id == AV\_CODEC\_ID\_AAC ||

track->par->codec\_id == AV\_CODEC\_ID\_AC3 ||

track->par->codec\_id == AV\_CODEC\_ID\_EAC3 ||

track->par->codec\_id == AV\_CODEC\_ID\_AMR\_NB ||

track->par->codec\_id == AV\_CODEC\_ID\_ALAC ||

track->par->codec\_id == AV\_CODEC\_ID\_ADPCM\_MS ||

track->par->codec\_id == AV\_CODEC\_ID\_ADPCM\_IMA\_WAV ||

track->par->codec\_id == AV\_CODEC\_ID\_QDM2 ||

(mov\_pcm\_le\_gt16(track->par->codec\_id) && version==1) ||

(mov\_pcm\_be\_gt16(track->par->codec\_id) && version==1)))

ret = mov\_write\_wave\_tag(s, pb, track);

else if (track->tag == MKTAG('m','p','4','a'))

ret = mov\_write\_esds\_tag(pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_AMR\_NB)

ret = mov\_write\_amr\_tag(pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_AC3)

ret = mov\_write\_ac3\_tag(s, pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_EAC3)

ret = mov\_write\_eac3\_tag(s, pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_ALAC)

ret = mov\_write\_extradata\_tag(pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_WMAPRO)

ret = mov\_write\_wfex\_tag(s, pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_FLAC)

ret = mov\_write\_dfla\_tag(pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_OPUS)

ret = mov\_write\_dops\_tag(s, pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_TRUEHD)

ret = mov\_write\_dmlp\_tag(s, pb, track);

else if (track->vos\_len > 0 **&& track->par->codec\_id != AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO**)

ret = mov\_write\_glbl\_tag(pb, track);

**Update #3:**

**Function:** static int mov\_write\_audio\_tag(AVFormatContext \*s, AVIOContext \*pb, MOVMuxContext \*mov, MOVTrack \*track)

**Line: 1339**

if (mov->write\_btrt &&

((ret = mov\_write\_btrt\_tag(pb, track)) < 0))

return ret;

**// To write the mhaC tag data for mp4 in case of AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO**

**if(track->par->codec\_id == AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO) {**

**mov\_write\_mhac\_tag(pb,track);**

**}**

ret = update\_size(pb, pos);

return ret;

**Update #4:**

**Function:** static int mov\_init(AVFormatContext \*s)

**Line: 7049**

for (i = 0; i < s->nb\_streams; i++) {

AVStream \*st= s->streams[i];

MOVTrack \*track= &mov->tracks[i];

AVDictionaryEntry \*lang = av\_dict\_get(st->metadata, "language", NULL,0);

track->st = st;

track->par = st->codecpar;

track->language = ff\_mov\_iso639\_to\_lang(lang?lang->value:"und", mov->mode!=MODE\_MOV);

if (track->language < 0)

track->language = 32767; // Unspecified Macintosh language code

track->mode = mov->mode;

**if(track->par->codec\_id == AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO)**

**{**

**// For MHA1**

**if((track->par->extradata)[track->par->extradata\_size - 1] == 2)**

**{**

**track->tag = MKTAG('m', 'h', 'a', '1');**

**}**

**// For MHM1**

**else**

**{**

**track->tag = MKTAG('m', 'h', 'm', '1');**

**}**

**}**

**else**

**{**

**track->tag = mov\_find\_codec\_tag(s, track);**

**}**

if (!track->tag) {

av\_log(s, AV\_LOG\_ERROR, "Could not find tag for codec %s in stream #%d, "

"codec not currently supported in container\n",

avcodec\_get\_name(st->codecpar->codec\_id), i);

return AVERROR(EINVAL);

}

**Update #5:**

**Function:** static const AVCodecTag codec\_mp4\_tags[]

**Line: 7751**

{ AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO, MKTAG('m', 'h', 'm', '1') },

**{ AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO, MKTAG('m', 'h', 'a', '1') },**

{ AV\_CODEC\_ID\_TTML, MOV\_MP4\_TTML\_TAG },

{ AV\_CODEC\_ID\_TTML, MOV\_ISMV\_TTML\_TAG },

{ AV\_CODEC\_ID\_NONE, 0 },

1. In *libavformat/isom\_tags.c* file:

**Update #6:**

**Function:** const AVCodecTag ff\_codec\_movaudio\_tags[]

**Line: 338**

{ AV\_CODEC\_ID\_OPUS, MKTAG('O', 'p', 'u', 's') }, /\* mp4ra.org \*/

{ AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO, MKTAG('m', 'h', 'm', '1') }, /\* MPEG-H 3D Audio bitstream \*/

**{ AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO, MKTAG('m', 'h', 'a', '1') },**

{ AV\_CODEC\_ID\_NONE, 0 },

## Steps for Configure, Build and Execute

1. Go to <ffmpeg> directory.
2. Configure ffmpeg using below command. No additional options are required for enabling Ittiam MPEGH Encoder.

./configure (and additional options…)

1. With successful execution of Step 2, <ffmpeg>/config\_components.h will be generated. Confirm CONFIG\_IA\_MPEGH\_ENCODER is set to 1 in <ffmpeg>/config\_components.h.
2. Build ffmpeg binaries using following command in <ffmpeg> folder:

make

After successful completion of make process, the *ffmpeg* executable with Ittiam MPEGH Encoder plugin will be ready to use.

For executing *ffmpeg* please take care of the following steps:

1. Ensure that shared object is present in the current directory (ffmpeg executable directory).
2. In case of production licenses, the license file to be present in the current directory (ffmpeg executable directory).
3. On Linux, before running *ffmpeg* plugin, path of the shared library should be set. <ffmpeg> path can be either absolute or relative.

Run the command export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:<ffmpeg>

1. On MAC, the path of shared object is already included in the binary which would be <ffmpeg> directory. However, if user wants to run the *ffmpeg* binary from some other directory, then, path of the shared library should be set. <ffmpeg> path can be either absolute or relative.

Run the command export DYLD\_LIBRARY\_PATH=$DYLD\_LIBRARY\_PATH:<ffmpeg>

# Integration of Plugin for Windows

## Prerequisite

Git needs to be installed in host PC. It can be downloaded from the below link:

<https://git-scm.com/downloads>.

## Creating DLL

FFMPEG requires encoder library in DLL format (.dll format). DLL can be created using the below steps:

* Download libmpeghe source code from <https://github.com/ittiam-systems/libmpeghe>.
* Copy the patch file libmpeghe/ffmpeg\_plugin/windows\_mpegh\_enc\_lib.patch to libmpeghe directory.
* Run the below command in git bash from libmpeghe directory to apply patch:

git apply windows\_mpegh\_enc\_lib.patch

* With patch applied, the source code files and the MSVS project file gets updated.
* Build the library using the MSVS solution file libmpeghe/build/msvs/impeghe\_lib.sln. The DLL can be created for x86 or x64 platform in both release and debug modes. For 64-bit host PC, build x64 release mode DLL.
* The DLL libia\_mpeghe.dll is generated in the libmpeghe/build/msvs/x64/Release directory.

## Plugin Integration to FFMPEG

The MPEG-H encoder plugin can be integrated into FFMPEG in two methods:

* Using the patch file
* Manual integration.

### Integration Using a Patch File

#### Applying the Patch

* Follow the steps mentioned in [section 3.2](#_Creating_DLL) if not done yet.
* Download the FFMPEG source code version 5.1 from <https://git.ffmpeg.org/ffmpeg.git> using the below commands in Git bash.

git clone <https://git.ffmpeg.org/ffmpeg.git>

cd ffmpeg

git checkout release/5.1

* Copy the patch file libmpeghe/ffmpeg\_plugin/ffmpeg\_mpegh\_enc.patch to ffmpeg directory.
* To apply the patch ffmpeg\_mpegh\_enc, use the command given below:

git apply ffmpeg\_mpegh\_enc.patch

* As the shared object libia\_mpeghe.dll is not part of the patch, it needs to be copied to ffmpeg folder manually.

Copy libia\_mpeghe.dllfrom libmpeghe/build/msvs/x64/Release *directory* toffmpeg/.

* Ittiam recommends referring https://trac.ffmpeg.org/wiki/CompilationGuide for further information on compilation and setup.
* Move to [Steps for Configure, Build and Execute](#_Steps_for_Configure,_1).

### Manual Integration

#### Steps for Integrating MPEGH Encoder Plugin into FFMPEG Framework Manually

1. Follow the steps mentioned in section 3.2 if not done yet.
2. Download the FFMPEG source code version 5.1 from <https://git.ffmpeg.org/ffmpeg.git> using the below commands in Git bash.

git clone <https://git.ffmpeg.org/ffmpeg.git>

cd ffmpeg

git checkout release/5.1

1. Copy *libia\_mpeghe.dll* from libmpeghe/build/msvs/x64/Releasedirectory toffmpeg/.
2. Copy all *C source* filesfrom libmpeghe*/*ffmpeg\_plugin*/ffmpeg/src/ to ffmpeg/libavcodec* folder.
3. Copy all *header* files from libmpeghe*/*ffmpeg\_plugin*/ffmpeg/include/* to *ffmpeg/libavcodec* folder.
4. Search for EXTRALIBS tag in configuration file (File name: *configure*) and pass the argument ‘-lia\_mpeghe -L.’. The command should look like as below.

EXTRALIBS=$extralibs **–lia\_mpeghe –L.**

This is for linking Ittiam MPEGH Encoder library.

1. Add the below line in file *ffmpeg/libavcodec/Makefile* under section ‘# decoders / encoders’.

**OBJS-$(CONFIG\_IA\_MPEGH\_ENCODER) += ia\_mpegh\_enc.o impeghe\_error.o**

1. Add the below line of code in the file *libavcodec/allcodecs.c* under section ‘*/\* external libraries \*/*’.

**extern const FFCodec ff\_ia\_mpegh\_encoder;**

Ittiam recommends referring <https://trac>.ffmpeg.org/wiki/CompilationGuide for further information on compilation and setup.

#### Steps for Adding MP4 Support

Below are the steps to add MPEGH format support to the existing FFMPEG’s MP4 multiplexer.

1. In *libavformat/movenc.c* file:

**Update #1**

**Line:1162 (**Add given function before static int mov\_write\_btrt\_tag(AVIOContext \*pb, MOVTrack \*track)**)**

**static int mov\_write\_mhac\_tag(AVIOContext \*pb, MOVTrack \*track)**

**{**

**// Size of mhaC**

**avio\_wb32(pb,track->par->extradata\_size + 8 -1);**

**ffio\_wfourcc(pb, “mhaC”);**

**// Writing the buffer**

**avio\_write(pb,track->par->extradata,track->par->extradata\_size – 1);**

**return 1;**

**}**

**Update #2**

**Function:** static int mov\_write\_audio\_tag(AVFormatContext \*s, AVIOContext \*pb, MOVMuxContext \*mov, MOVTrack \*track)

**Line: 1293**

if (track->mode == MODE\_MOV &&

(track->par->codec\_id == AV\_CODEC\_ID\_AAC ||

track->par->codec\_id == AV\_CODEC\_ID\_AC3 ||

track->par->codec\_id == AV\_CODEC\_ID\_EAC3 ||

track->par->codec\_id == AV\_CODEC\_ID\_AMR\_NB ||

track->par->codec\_id == AV\_CODEC\_ID\_ALAC ||

track->par->codec\_id == AV\_CODEC\_ID\_ADPCM\_MS ||

track->par->codec\_id == AV\_CODEC\_ID\_ADPCM\_IMA\_WAV ||

track->par->codec\_id == AV\_CODEC\_ID\_QDM2 ||

(mov\_pcm\_le\_gt16(track->par->codec\_id) && version==1) ||

(mov\_pcm\_be\_gt16(track->par->codec\_id) && version==1)))

ret = mov\_write\_wave\_tag(s, pb, track);

else if (track->tag == MKTAG(‘m’,’p’,’4’,’a’))

ret = mov\_write\_esds\_tag(pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_AMR\_NB)

ret = mov\_write\_amr\_tag(pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_AC3)

ret = mov\_write\_ac3\_tag(s, pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_EAC3)

ret = mov\_write\_eac3\_tag(s, pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_ALAC)

ret = mov\_write\_extradata\_tag(pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_WMAPRO)

ret = mov\_write\_wfex\_tag(s, pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_FLAC)

ret = mov\_write\_dfla\_tag(pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_OPUS)

ret = mov\_write\_dops\_tag(s, pb, track);

else if (track->par->codec\_id == AV\_CODEC\_ID\_TRUEHD)

ret = mov\_write\_dmlp\_tag(s, pb, track);

else if (track->vos\_len > 0 **&& track->par->codec\_id != AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO**)

ret = mov\_write\_glbl\_tag(pb, track);

**Update #3:**

**Function:** static int mov\_write\_audio\_tag(AVFormatContext \*s, AVIOContext \*pb, MOVMuxContext \*mov, MOVTrack \*track)

**Line: 1339**

if (mov->write\_btrt &&

((ret = mov\_write\_btrt\_tag(pb, track)) < 0))

return ret;

**// To write the mhaC tag data for mp4 in case of AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO**

**if(track->par->codec\_id == AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO) {**

**mov\_write\_mhac\_tag(pb,track);**

**}**

ret = update\_size(pb, pos);

return ret;

**Update #4:**

**Function:** static int mov\_init(AVFormatContext \*s)

**Line: 7049**

for (I = 0; I < s->nb\_streams; i++) {

AVStream \*st= s->streams[i];

MOVTrack \*track= &mov->tracks[i];

AVDictionaryEntry \*lang = av\_dict\_get(st->metadata, “language”, NULL,0);

track->st = st;

track->par = st->codecpar;

track->language = ff\_mov\_iso639\_to\_lang(lang?lang->value:”und”, mov->mode!=MODE\_MOV);

if (track->language < 0)

track->language = 32767; // Unspecified Macintosh language code

track->mode = mov->mode;

**if(track->par->codec\_id == AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO)**

**{**

**// For MHA1**

**if((track->par->extradata)[track->par->extradata\_size – 1] == 2)**

**{**

**track->tag = MKTAG(‘m’, ‘h’, ‘a’, ‘1’);**

**}**

**// For MHM1**

**else**

**{**

**track->tag = MKTAG(‘m’, ‘h’, ‘m’, ‘1’);**

**}**

**}**

**else**

**{**

**track->tag = mov\_find\_codec\_tag(s, track);**

**}**

if (!track->tag) {

av\_log(s, AV\_LOG\_ERROR, “Could not find tag for codec %s in stream #%d, “

“codec not currently supported in container\n”,

avcodec\_get\_name(st->codecpar->codec\_id), i);

return AVERROR(EINVAL);

}

**Update #5:**

**Function:** static const AVCodecTag codec\_mp4\_tags[]

**Line: 7751**

{ AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO, MKTAG(‘m’, ‘h’, ‘m’, ‘1’) },

**{ AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO, MKTAG(‘m’, ‘h’, ‘a’, ‘1’) },**

{ AV\_CODEC\_ID\_TTML, MOV\_MP4\_TTML\_TAG },

{ AV\_CODEC\_ID\_TTML, MOV\_ISMV\_TTML\_TAG },

{ AV\_CODEC\_ID\_NONE, 0 },

1. In *libavformat/isom\_tags.c* file:

**Update #6:**

**Function:** const AVCodecTag ff\_codec\_movaudio\_tags[]

**Line: 338**

{ AV\_CODEC\_ID\_OPUS, MKTAG('O', 'p', 'u', 's') }, /\* mp4ra.org \*/

{ AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO, MKTAG('m', 'h', 'm', '1') }, /\* MPEG-H 3D Audio bitstream \*/

**{ AV\_CODEC\_ID\_MPEGH\_3D\_AUDIO, MKTAG('m', 'h', 'a', '1') },**

{ AV\_CODEC\_ID\_NONE, 0 },

## Steps for Configure, Build and Execute

1. Open MSYS2 UCRT64 terminal. It is assumed that all dependencies are installed as per FFMPEG Compilation guide instructions.
2. Go to <ffmpeg> directory.
3. Configure ffmpeg using below command. No additional options are required for enabling Ittiam MPEGH Encoder.

./configure (and additional options…)

1. With successful execution of Step 3, <ffmpeg>/config\_components.h will be generated. Confirm CONFIG\_IA\_MPEGH\_ENCODER is set to 1 in <ffmpeg>/config\_components.h.
2. Build ffmpeg binaries using following command in <ffmpeg> folder.

make

After successful completion of make process, the *ffmpeg* executable with Ittiam MPEGH Encoder plugin will be ready to use.

For executing *ffmpeg* please take care of the following steps:

1. Ensure that DLL is present in the current directory (ffmpeg executable directory).
2. In case of production licenses, the license file to be present in the current directory (ffmpeg executable directory).

# Usage of Plugin

## Command Line Arguments

**-ia\_bit\_rate <bit\_rate>**: Bit Rate. Default Value: 64000, Range: 32000 to 512000.

**-ia\_op\_fmt <output\_format>**: Output Format. Default Value: 1 (RAW\_MHAS). Supported values: 1 (RAW\_MHAS), 2 (MP4\_MHA1) and 3 (MP4\_MHM1).

**-ia\_cicp <cicp\_index>**: Channel Configuration Index. Default Value: 1. Range: 1 to 20 except 8. Refer to [ISO - ISO/IEC 23091-3:2018 - Information technology — Coding-independent code points — Part 3: Audio](https://www.iso.org/standard/73413.html) for more information.

**-ia\_chans <no\_of\_channels>**: Number of Channels. Default Value: 1. Range: 1 to 24.

**-ia\_sample\_rate <sample\_rate>**: Sample Rate. Default Value: 48000. Range: 14700 to 48000.

**-ia\_pcm\_sz <pcm\_word\_size>:** PCM Word Size. Default Value: 16. Supported values: 16, 24 and 32.

**-ia\_oam\_file <oam\_file\_name>:** OAM File. Default Value: NULL.

**-ia\_hoa\_file <first hoa filename that ends with 00+.wav>:** First HOA Filename that ends with 00+.wav. Default Value: NULL.

Refer to <https://github.com/ittiam-systems/libmpeghe/blob/main/README.md> for more information about default values and range of various parameters.

## Command Usage

### For MHAS Output Format

In the git bash (for windows) or in the terminal (Linux/MAC), below commands can be run for encoding the file into MPEG-H stream.

* **Without OAM or HOA File**

./ffmpeg -i <input\_file\_name>.wav -f s16le -c:a ia\_mpeghe -ia\_bit\_rate <bit\_rate> -ia\_pcm\_sz <pcm\_word\_size> -ia\_sample\_rate <sample\_rate> -ia\_cicp <cicp\_index> -ia\_chans <no\_of\_channels> -ia\_op\_fmt 1 <output\_file\_name>.mhas

Example:

# 2 channel

./ffmpeg -i Test.wav -f s16le -c:a ia\_mpeghe -ia\_bit\_rate 64000 -ia\_op\_fmt 1 -ia\_chans 2 -ia\_cicp 2 -ia\_sample\_rate 48000 -ia\_pcm\_sz 16 Test\_enc.mhas

* **With OAM File**

./ffmpeg -i <path>/<input\_file\_name>.wav -f s16le -c:a ia\_mpeghe   
-ia\_bit\_rate <bit\_rate> -ia\_pcm\_sz <pcm\_word\_size> -ia\_sample\_rate <sample\_rate> -ia\_cicp <cicp\_index> -ia\_op\_fmt 1 -ia\_oam\_file <path>/<oam\_file\_name> <output\_file\_name>.mhas

Example:

./ffmpeg -i ./Test\_000.wav -f s16le -c:a ia\_mpeghe -ia\_op\_fmt 1 -ia\_cicp 6 -ia\_bit\_rate 256000 -ia\_sample\_rate 48000 -ia\_oam\_file ./Test.oam Test\_OAM\_enc.mhas

* **With HOA File**

./ffmpeg -i <first hoa filename that ends with 00+.wav>.wav -f s16le -c:a ia\_mpeghe -ia\_bit\_rate <bit\_rate> -ia\_pcm\_sz <pcm\_word\_size> -ia\_sample\_rate <sample\_rate> -ia\_cicp <cicp\_index> -ia\_op\_fmt 1 -ia\_hoa\_file <first hoa filename that ends with 00+.wav> <output\_file\_name>.mhas

Example:

# HOA - 3rd Order example

./ffmpeg -i Test3/test\_3\_00+.wav -f s16le -c:a ia\_mpeghe -ia\_op\_fmt 1   
-ia\_cicp 6 -ia\_bit\_rate 512000 -ia\_sample\_rate 48000 -ia\_pcm\_sz 32   
-ia\_hoa\_file Test3/test\_3\_00+.wav test\_3\_HOA\_enc.mhas

### For MP4 Output Format

FFMPEG, by default outputs MP4 header at the end of file. “–movflags faststart” forces FFMPEG to output MP4 header at the beginning of the file.

* **Without OAM or HOA File**

./ffmpeg -i <input\_file\_name>.wav -movflags faststart -c:a ia\_mpeghe   
-ia\_bit\_rate <bit\_rate> -ia\_pcm\_sz <pcm\_word\_size> -ia\_sample\_rate <sample\_rate> -ia\_cicp <cicp\_index> -ia\_chans <no\_of\_channels> -ia\_op\_fmt <output\_format> <output\_file\_name>.mp4

Example:

# 2 channel

./ffmpeg -i Test.wav -movflags faststart -c:a ia\_mpeghe -ia\_bit\_rate 64000 -ia\_op\_fmt 2 -ia\_chans 2 -ia\_cicp 2 -ia\_sample\_rate 48000 -ia\_pcm\_sz 16 Test\_enc.mp4

* **With OAM File**

./ffmpeg -i <path>/<input\_file\_name>.wav -movflags faststart -c:a ia\_mpeghe -ia\_bit\_rate <bit\_rate> -ia\_pcm\_sz <pcm\_word\_size> -ia\_sample\_rate <sample\_rate> -ia\_cicp <cicp\_index> -ia\_op\_fmt <output\_format> –ia\_oam\_file <path>/<oam\_file\_name> <output\_file\_name>.mp4

Example:

./ffmpeg -i ./Test\_000.wav -movflags faststart -c:a ia\_mpeghe -ia\_op\_fmt 2 -ia\_cicp 6 -ia\_bit\_rate 256000 -ia\_sample\_rate 48000 -ia\_oam\_file ./Test.oam Test\_OAM\_enc.mp4

* **With HOA File**

./ffmpeg -i <first hoa filename that ends with 00+.wav>.wav -movflags faststart -c:a ia\_mpeghe -ia\_bit\_rate <bit\_rate> -ia\_pcm\_sz <pcm\_word\_size> -ia\_sample\_rate <sample\_rate> -ia\_cicp <cicp\_index> -ia\_op\_fmt <output\_format> -ia\_hoa\_file <first hoa filename that ends with 00+.wav> <output\_file\_name>.mp4

Example:

# HOA - 3rd Order example

./ffmpeg -i Test3/test\_3\_00+.wav -movflags faststart -c:a ia\_mpeghe   
-ia\_op\_fmt 2 -ia\_cicp 6 -ia\_bit\_rate 512000 -ia\_sample\_rate 48000 -ia\_pcm\_sz 32 -ia\_hoa\_file Test3/test\_3\_00+.wav test\_3\_HOA\_enc.mp4

# Notes and Limitations

* For channels-only input case (where inputs involve individual MONO channel files), merge all the input files into a single multi-channel file using SOX utility. The convention for multi-channel files is based on CICP and further details can be found at [ISO - ISO/IEC 23091-3:2018 - Information technology — Coding-independent code points — Part 3: Audio](https://www.iso.org/standard/73413.html). This multi-channel file can be used as input for the encoder.

SoX utility in Linux can be installed using below command:

apt-get install sox

SoX utility in Windows can be installed using below link:

[SoX - Sound eXchange download | SourceForge.net](https://sourceforge.net/projects/sox/)

* If OAM file is given as input, then while giving the input flag (***-i***) with FFMPEG, all the object and channel files should be of same size. The naming convention for OAM use-case and OAM file format details can be found in [w13412 (3D\_EncoderInputFormat).docx](https://mpeg.chiariglione.org/standards/mpeg-h/3d-audio/encoder-input-format-mpeg-h-3d-audio).

For OAM file case,

Channel files should be named in the format *item\_name\_A<azimuth\_angle>\_E<elevation\_angle>.wav* and be placed in the same folder as the *.oam* file.

Example: *item\_name\_A+030\_E+00.wav*

Object files should be named as “*<item\_name>\_<object\_idx>.wav*” and be placed in the same folder as the *.oam* file. The *<object\_idx>* is a three-digit number counted from zero (padded from the left with zeros if needed).

Example: *item\_name\_000.wav*

Also, for multiple object files case, “***-i***” option requires only the first object file (with path) that ends with “*\_000.wav*”. The rest of the details will be fetched by the plugin from the *.oam* file. The object and channel files shall not be combined and be used as individual files only.

* If HOA file is given, then while giving the input flag (***-i***) with FFMPEG, all the input files need to be of same size. The naming convention of each HOA channel signal need to adhere to Section 4.1 of [w13412 (3D\_EncoderInputFormat).docx](https://mpeg.chiariglione.org/standards/mpeg-h/3d-audio/encoder-input-format-mpeg-h-3d-audio).

The “***-i***” option and “***-ia\_hoa\_file***” requires only the first HOA file name that ends with “*00+.wav”*. The plugin fetches the rest of the file names itself internally since the files are named as per convention. The individual HOA input files (ACN/SN3D input files) shall be used as-is and shall not be combined into a single file.

* For handling HOA case, the argument “***-ia\_hoa\_file***” has been added. However, FFMPEG mandatorily requires “***-i***” option and it will give input to the encoder module only upto the size of file mentioned along with “***-i***” option. Hence, for HOA case, there appears like two options need to be given for input files.
* For 24-bit input wave files, FFMPEG appends one byte of extra zeros. Hence, for 24-bit input files, when encoding the files in MHAS format, “***-ia\_pcm\_sz 32***” option needs to be used.
* In this document, the command line options are used along with a “minus (-)” sign. The document might show up as a different symbol, hence, it needs to be ensured that the (-) sign only is used with all options when trying the FFMPEG pipeline.
* Whenever the output is generated in mhas format, i.e. when the output file name is given the extension of *.mhas*, FFMPEG is expecting the option (***-f***) to be given without which an error “*unable to find a suitable output format*” is reported. The option ***-f*** does not have any impact on encoding. This option is not needed if the output format is MP4.
* On few Linux PCs, when patch was applied, “bad-interpreter” error was encountered due to (CTRL+M) character. If this issue is observed, edit “configure” file manually (one line change) as per the steps mentioned under [Manual Integration](#_Manual_Integration).
* The files generated using the plugin can be decoded using the decoder available at <https://github.com/ittiam-systems/libmpegh>. This decoder may throw error if the encoded stream is of unsupported profile.