Document Number: UM0097

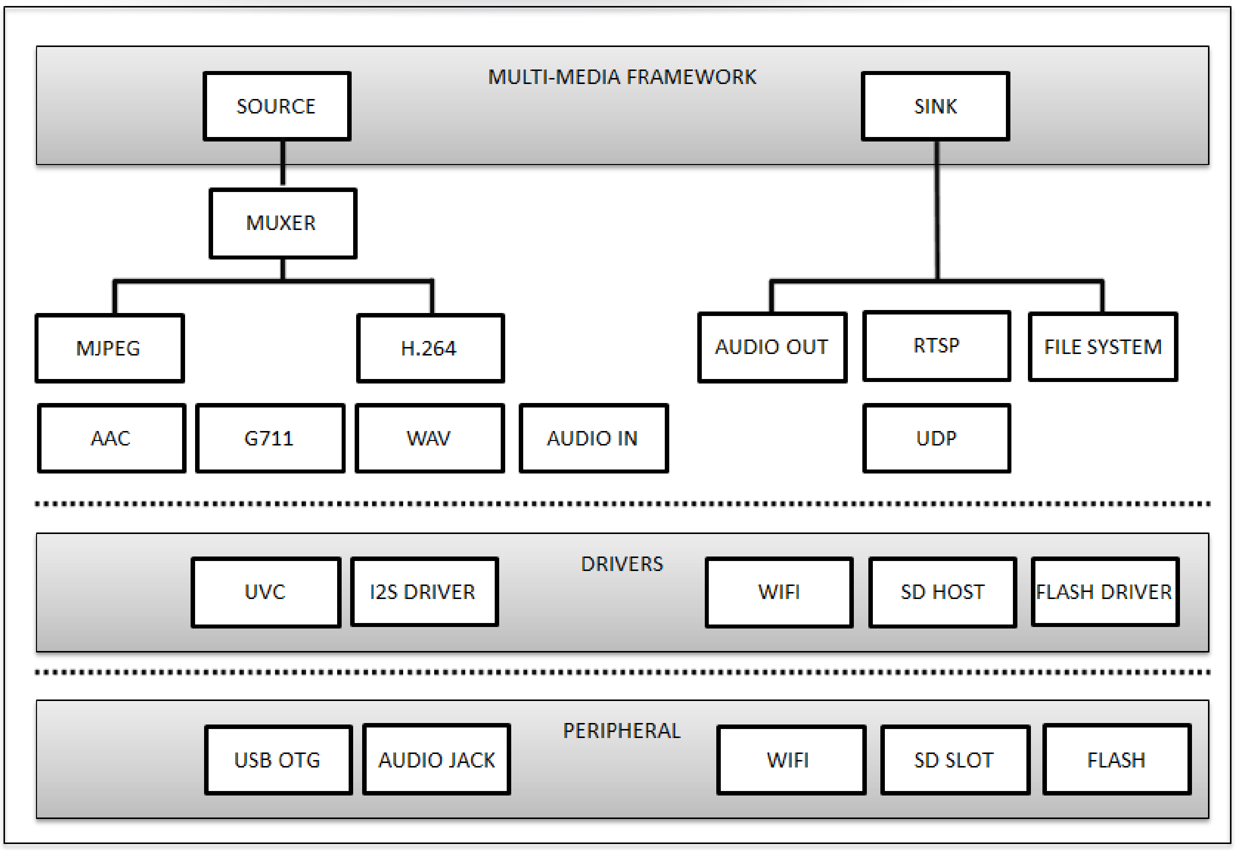
Multimedia framework user manual

The document describe how to use the multimedia framework

Table of Contents

1 Introduction Multi Media Framework (MMF)

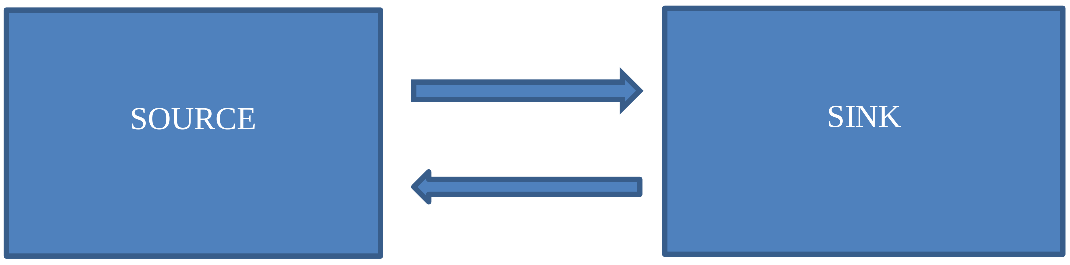
The Multimedia Framework (MMF) is used to handle all sorts of media applications on Ameba. The MMF structure is as shown in the following chart.



There are two important entities in the MMF. One is the source that produces the resource and the other is sink that consumes the resource. The source can be the file input, UVC camera, or storage; the sink can be RTSP or other stream.

In order to use the MMF, the following aspects must be fulfilled.

- Define a valid source
- Define a valid sink
- Create OS message queues in application level for both source and sink module to access. Module-specific configuration may be made if required.
- Call the MMF APIs to start the multimedia streaming.



2 Implement source and sink

MMF allows users to define our own source and sink modules depending upon the application. Although implementation details may be different, these basic rules of the MMF structure should be a little bit similar. This section shall explain how to create a source and a sink in MMF using the example UVC source and RTSP sink. Further explanations on how to run the other examples are discussed in Section 3.

2.1 Media Source and Sink Module

MMF requires users to predefine both source and sink modules that implement create, destroy, set_param, and handle function callbacks.

Media Source Module	**Media Sink Module**

typedef struct _media_source_module{
void* (*create)(void);
void (*destroy)(void*);
int (*set_param)(void*, int, int);
int (*handle)(void*, void*);
}msrc_module_t;
	typedef struct _media_sink_module{
	void* (*create)(void);
	void (*destroy)(void*);
	int (*set_param)(void*, int, int);
	int (*handle)(void*, void*);
	}msink_module_t;

- (*create): pointer to the function that loads and initializes the source/sink stream of the module. For example, for UVC source, it points to the function in which the UVC driver is initialized and the corresponding context is returned.
- (*destroy): pointer to the function that de-initializes module instance and releases resource. For example, for UVC source, it points to function in which UVC driver is de-initialized and the corresponding context is released.
- (*set_param): pointer to function that sends the control command to the MMF layer or a specific module. For example, for UVC source, it points to function that controls UVC parameters (“frame height”, “frame width”, “framerate”, etc.) and MMF service task on/off.
- (*handle): pointer to the function that manipulates media data (how to produce data in source or how to consume data in sink). Data is transferred from source to sink and vice versa by means of OS message queue. Please note that MMF service task reacts differently based on message exchange buffer status. The status codes are introduced as follows:

```
o   #define STAT_INIT        0    //indicating buffer using at first time
o   #define STAT_USED        1    //indicating buffer consumed by sink
                                  //(must update this status in sink handle)
o   #define STAT_READY       2    //indicating buffer ready for use by source
                                  //(must update this status in source handle)
o   #define STAT_RESERVED    3    //indicating buffer being processed or
                                  //reserved and can’t be accessed right now
```

After msrc_module_t and msink_module_t structure are properly defined, user should declare the customized source and sink modules in mmf_source_list.h and mmf_sink_list.h, respectively. Thus, MMF can be aware of new-defined modules and these customized modules can be executed. Some built-in source and sink module definitions can be found under the path 'component\common\media\framework'.

2.2 MMF API Documentation

This section discusses the usage of MMF APIs.

msrc_context* mmf_source_open(msrc_module_t* source)
Create MMF source context instance. msrc_module_t* source: specify which source module class to be created. Return value: a pointer to source context instance.

void mmf_source_close(msrc_context* ctx)
Release the resource related to source context instance and lower level memory. msrc_context* ctx: pointer to source context instance that is ready to be released.

int mmf_source_ctrl(msrc_context* ctx, int cmd, int arg)
Set control command to a specific source context instance. msrc_context* ctx: pointer to a specific source context instance. int cmd: the input command code, needs to be defined in mmf_common.h. int arg: relevant value along with the input command code. Pass 0 if not applicable.

int mmf_source_get_frame(msrc_context* ctx, exch_buf_t *exbuf)
Prepare one frame or block of data and add its reference to message exchange buffer. msrc_context* ctx: pointer to a specific source context instance. exch_buf_t *exbuf: pointer to the message exchange buffer used to convey media frame.

msink_context* mmf_sink_open(msink_module_t *sink)
The functionality is similar to "mmf_source_open".

void mmf_sink_close(msink_context* ctx)
The functionality is similar to "mmf_source_close".

int mmf_sink_ctrl(msink_context* ctx, int cmd, int arg)
The functionality is similar to "mmf_source_ctrl".

int mmf_sink_put_frame(msink_context* ctx, exch_buf_t* exbuf)
Dereference one frame or block of data from message exchange buffer. msink_context* ctx: pointer to a specific sink context instance.

exch_buf_t *exbuf: pointer to the message exchange buffer used to convey media frame.

2.3 MP3 API Documentation

This section discusses the usage of MP3 APIs.

mp3_decoder_t mp3_create(void);

Creates the mp3 decoder instance and allocates required memory.
Input Parameter: No input parameters required
Return value: a pointer to the mp3_decoder_t structure. This must be created locally in the application before calling the create function.

int mp3_decode(mp3_decoder_t *dec, void *buf, int bytes, signed short *out, mp3_info_t *info);

After creation of the mp3 decoder instance use this API to initiate the actual decoding process.
Input Parameters:
→mp3_decoder_t* dec:-Pass the mp3 decoder instance that was obtained using the create function mentioned before.
→void* buf:- The buffer pointer to the input frame of the mp3 data that needs to be decoded.
→int bytes:- The number of bytes in the input mp3 data buffer.
→signed short *out:- Pointer to the buffer where the decoded mp3 data will be present after the decoding process is over.
→mp3_info_t *info:- This structure is used to identify the frequency of the decoded mp3 data and number of channels and the number of decoded bytes in each frame of the output data, details regarding this structure is found in the g711_decoder.h file.

Return Value:
Int, this integer returns the actual number of bytes that were successfully decoded from the input buffer provided to the mp3 decode function, based on this value the part of the input buffer that was not decoded must be provided again in the next iteration for decoding.

void mp3_done(mp3_decoder_t *dec);

Release the memory allocated for the mp3 instance and delete the mp3 instance.
Input Parameters:
mp3_decoder_t* dec :- The original instance created using mp3_create needs to be passed in as the input to this parameter.
Output Parameter:
None

2.4 Mp3 Config Table Detail and supported frequencies

In order to use the Mp3 decoder and hear the output audio in a smooth and uninterrupted manner the I2S driver must be configured with the right output frequency, the number of channels in the audio data and the I2S DMA page size must match the exact number of decoded bytes per frame of the mp3 in order to hear clear and uninterrupted audio from the I2S output.

2.4.1 Key Properties of an Mp3 File

- **Bit-Rate**
  This determines the number of bits contained in one second of the mp3 file. The bit rate of the mp3 file determines the size of each input frame. The input frame is decoded by the mp3 decoder to produce the output buffer that contains the raw audio data.
- **Frequency**
  This determines the sampling frequency of the input audio signal. This is an important parameter that determines the size of the decoded output frame. In case the audio is being played by means of I2S, the I2S driver must be initialized to the right frequency in order to be able to hear the audio clearly.
- **Number of Channels**
  An mp3 audio file can have one or two channels; this parameter determines the audio that needs to be played in the left and right channel respectively.

  The mp3 audio data properties can be checked using any media player. Most common media players like vlc and Audacity can be used to check the frequency and number of channels present in the mp3 file.

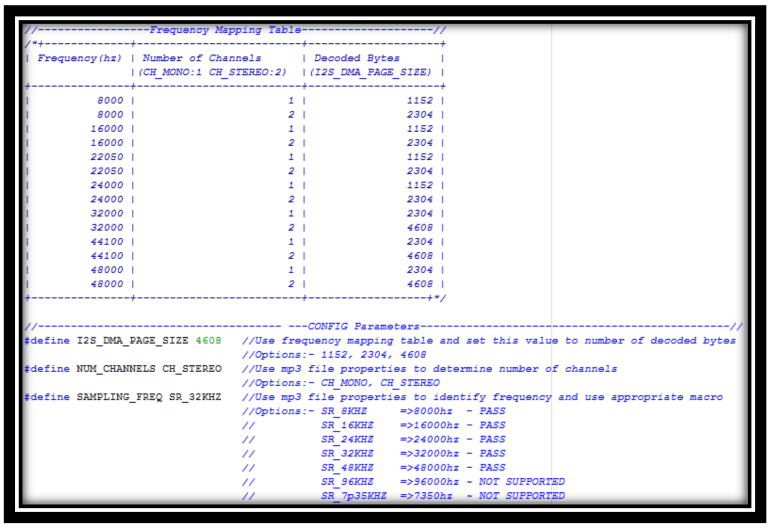
2.4.2 Mp3 Frequency table

This frequency table helps map the frequency and the number of channels in the mp3 files to the size of the output buffer that needs to be used to accumulate the decoded output of the mp3 decoder. The frequencies supported by the mp3 decoder are listed below.

Frequency(hz)	Channels	Decoded Bytes
8000	1	1152
8000	2	2304
16000	1	1152
16000	2	2304
22050	1	1152
22050	2	2304
24000	1	1152

24000	2	2304
32000	1	2304
32000	2	4608
44100	1	2304
44100	2	4608
48000	1	2304
48000	2	4608

In order to hear clear audio from the examples, the I2S driver should be initialized to the right frequency and the I2S DMA page size must be set to the value of decoded bytes by using the table shown above. These details are provided in the start of the mp3 example as well as shown below.



```
//----------------Frequency Mapping Table-------------------//
/*+-------------+-----------------------+------------------+
| Frequency(hz) | Number of Channels     | Decoded Bytes    |
                |(CH_MONO:1 CH_STEREO:2) |(I2S_DMA_PAGE_SIZE) |
+-------------+-----------------------+------------------+
|          8000 |                      1 |             1152 |
|          8000 |                      2 |             2304 |
|         16000 |                      1 |             1152 |
|         16000 |                      2 |             2304 |
|         22050 |                      1 |             1152 |
|         22050 |                      2 |             2304 |
|         24000 |                      1 |             1152 |
|         24000 |                      2 |             2304 |
|         32000 |                      1 |             2304 |
|         32000 |                      2 |             4608 |
|         44100 |                      1 |             2304 |
|         44100 |                      2 |             4608 |
|         48000 |                      1 |             2304 |
|         48000 |                      2 |             4608 |
+-------------+-----------------------+-----------------+*/

//------------------------------------ ---CONFIG Parameters-----------------------------------------//
#define I2S_DMA_PAGE_SIZE 4608   //Use frequency mapping table and set this value to number of decoded bytes
                                 //Options:- 1152, 2304, 4608
#define NUM_CHANNELS CH_STEREO   //Use mp3 file properties to determine number of channels
                                 //Options:- CH_MONO, CH_STEREO
#define SAMPLING_FREQ SR_32KHZ   //Use mp3 file properties to identify frequency and use appropriate macro
                                 //Options:- SR_8KHZ     =>8000hz  - PASS
                                 //          SR_16KHZ    =>16000hz - PASS
                                 //          SR_24KHZ    =>24000hz - PASS
                                 //          SR_32KHZ    =>32000hz - PASS
                                 //          SR_48KHZ    =>48000hz - PASS
                                 //          SR_96KHZ    =>96000hz ~ NOT SUPPORTED
                                 //          SR_7p35KHZ  =>7350hz  ~ NOT SUPPORTED
```

How to Run these Examples

The following sections provide the examples of using the MMF and can be used as references to create customized applications.

2.5 Media Single Stream – Source UVC Camera, Sink RTSP Stream

Please refer to following steps to understand how to run the MMF example with source as UVC camera and sink as RTSP stream.

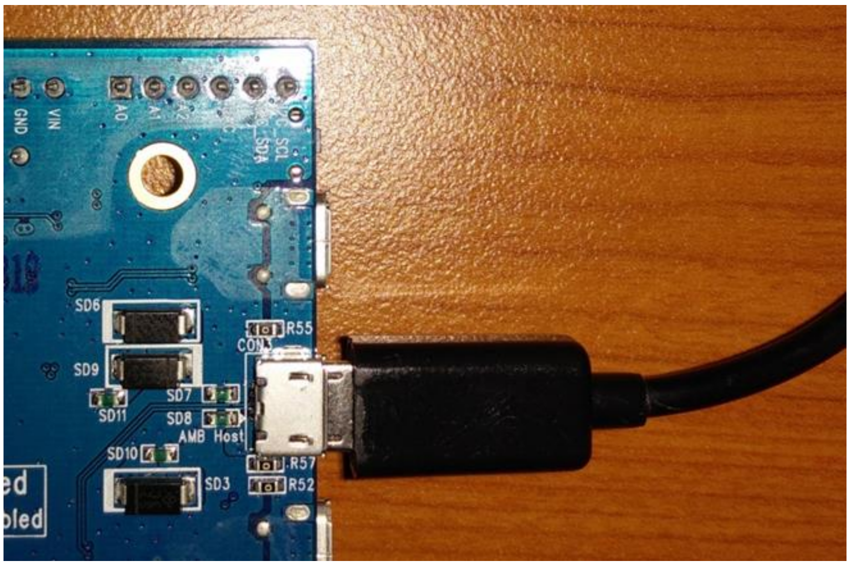
<u>Pre-requisites</u>

- Ameba board
- UVC camera (E.g., Logitech C170)
- USB cable is used to connect with PC.

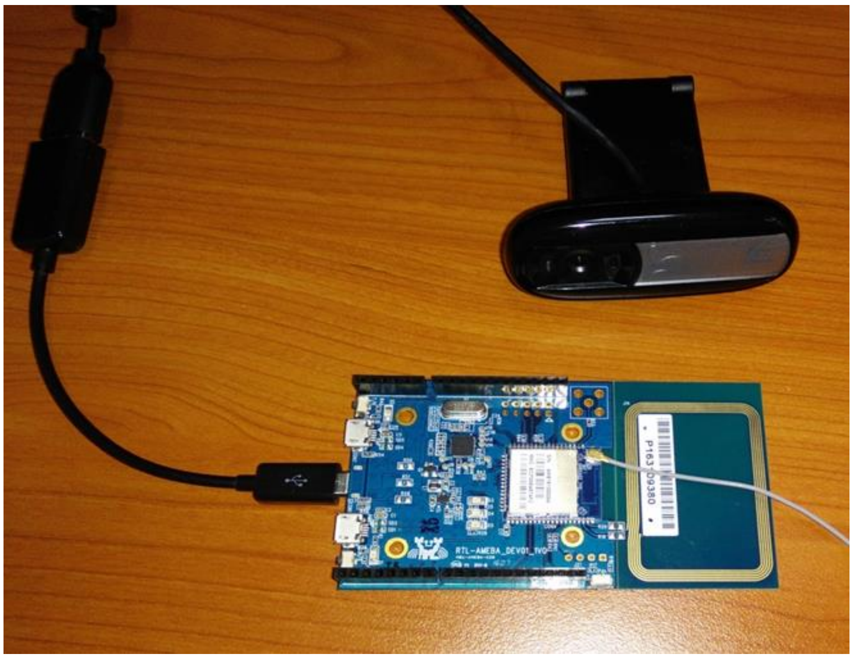
- WiFi connection for sending RTSP streaming out
- USB OTG cable is used to connect UVC camera to Ameba board.

Hardware Setup

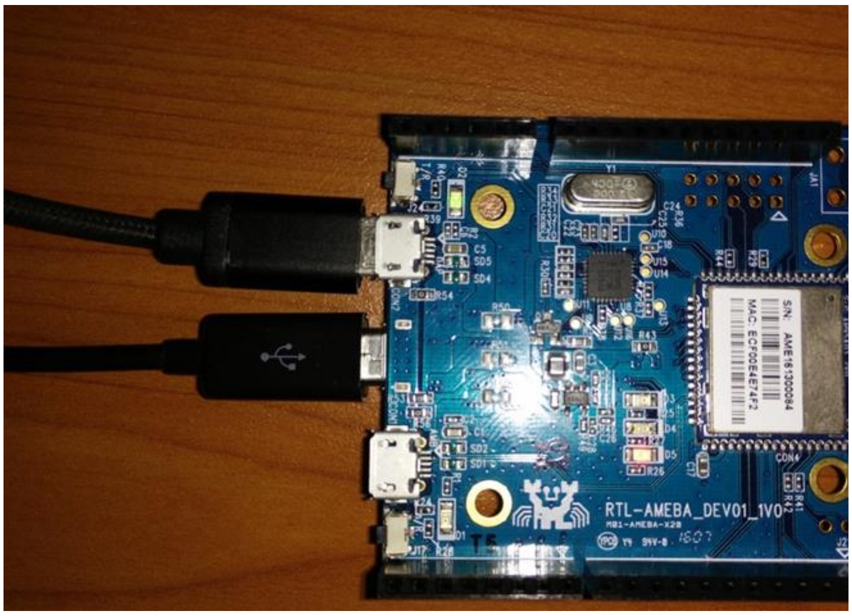
- Connect the mini-USB OTG cable to the port, labelled as “con3”, on the underside of the Ameba board as shown in the following figure.



- Next, connect the USB camera to the mini-USB OTG cable which is connected with the Ameba board as shown in the following figure.

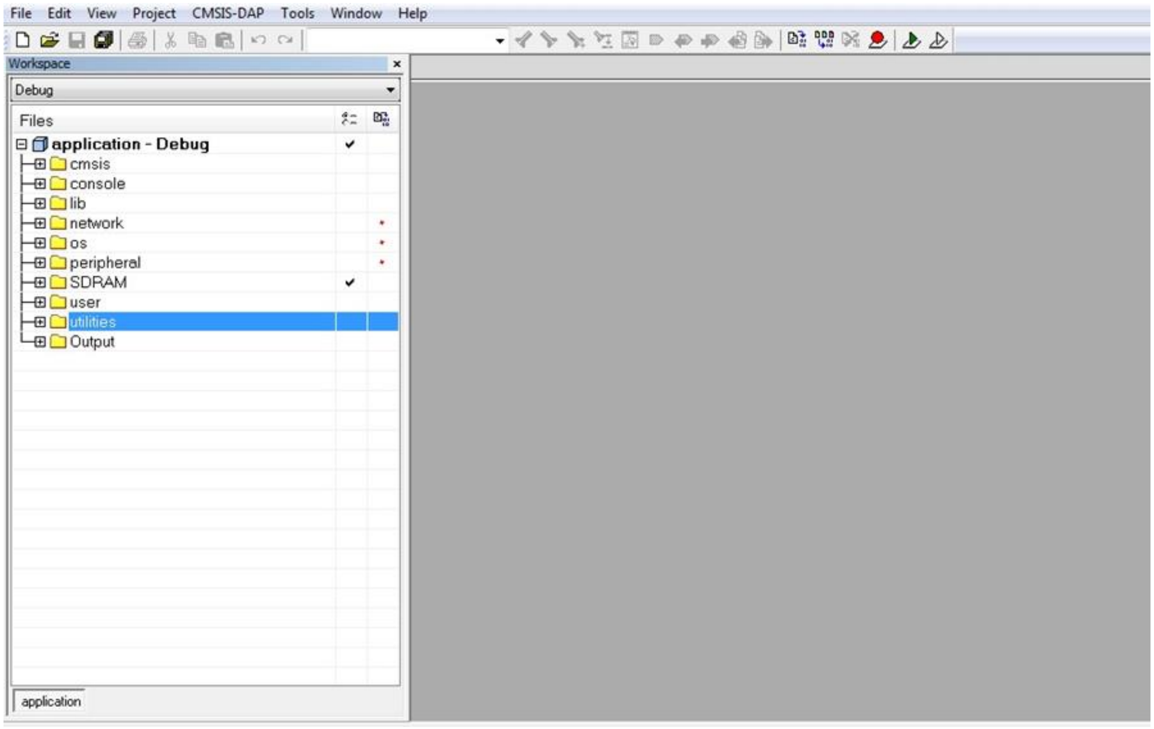


- Finally, connect the standard micro-USB cable to the port, labelled as “con1”, on the upper side of the Ameba board, as shown in the figure bellow, and connect the other end to the PC.

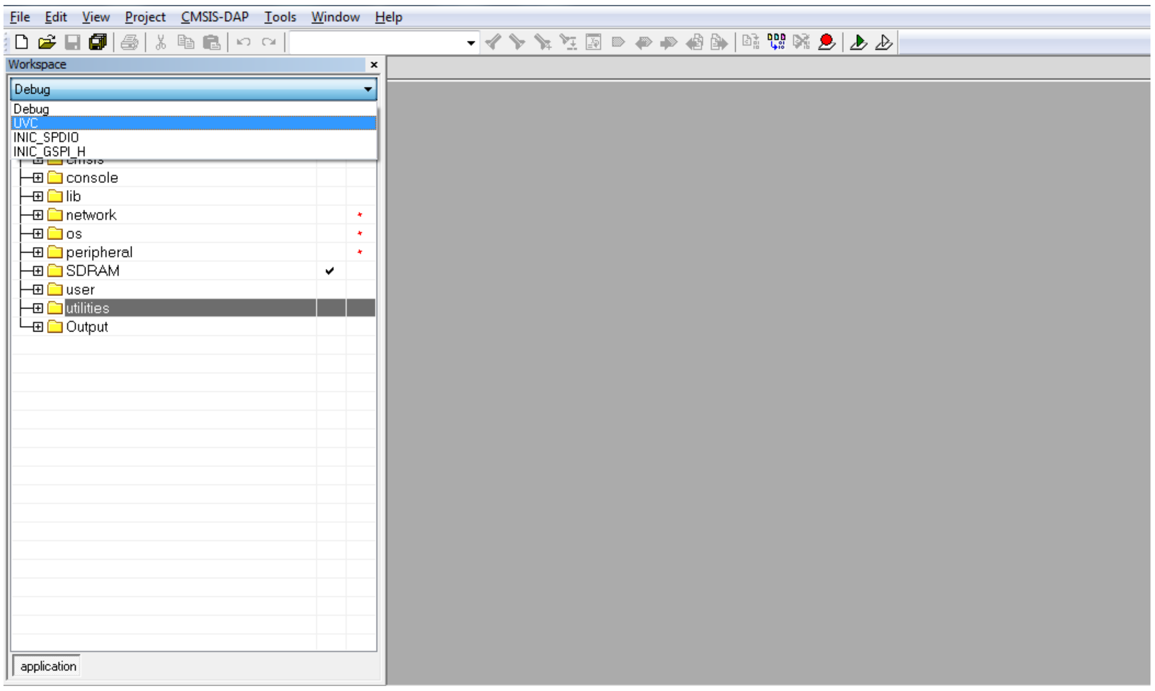


Software Setup

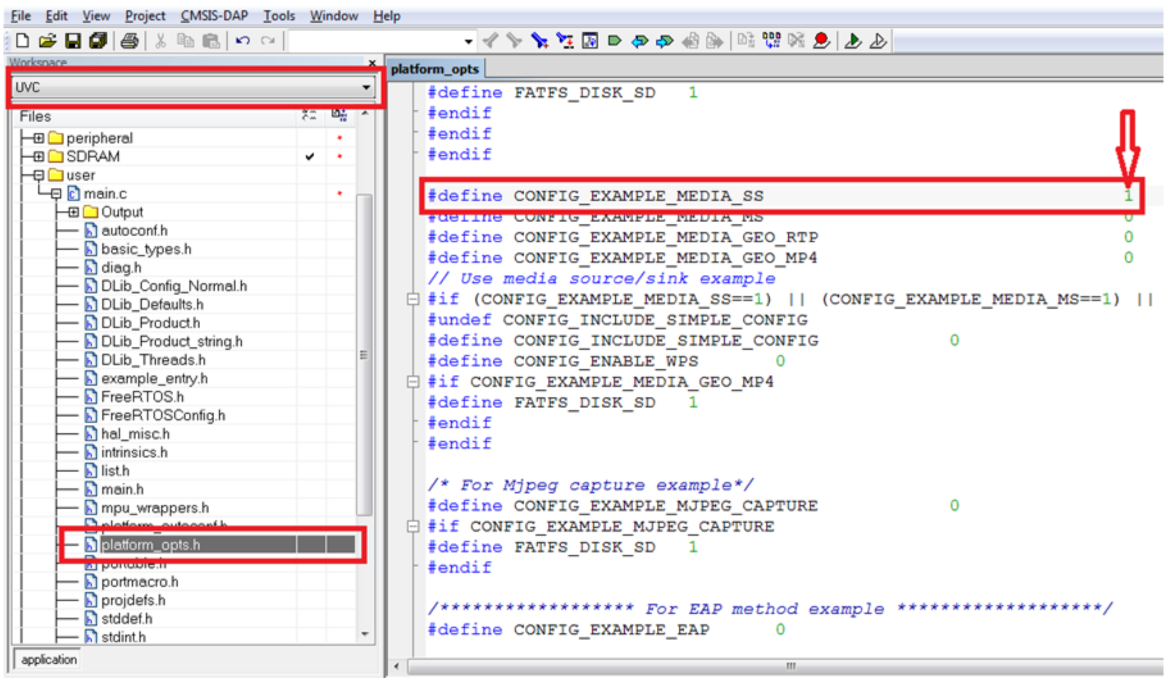
- Once the hardware setup is completed as mentioned above, please refer to the software setup steps in order to run the example. First, please open the IAR workspace, Project.eww in EWARM-RELEASE folder, and then the default workspace, Debug, is loaded as shown in the following figure.



- In order to use the UVC features, please select the UVC workspace. In the list box, please select the UVC option as shown below.

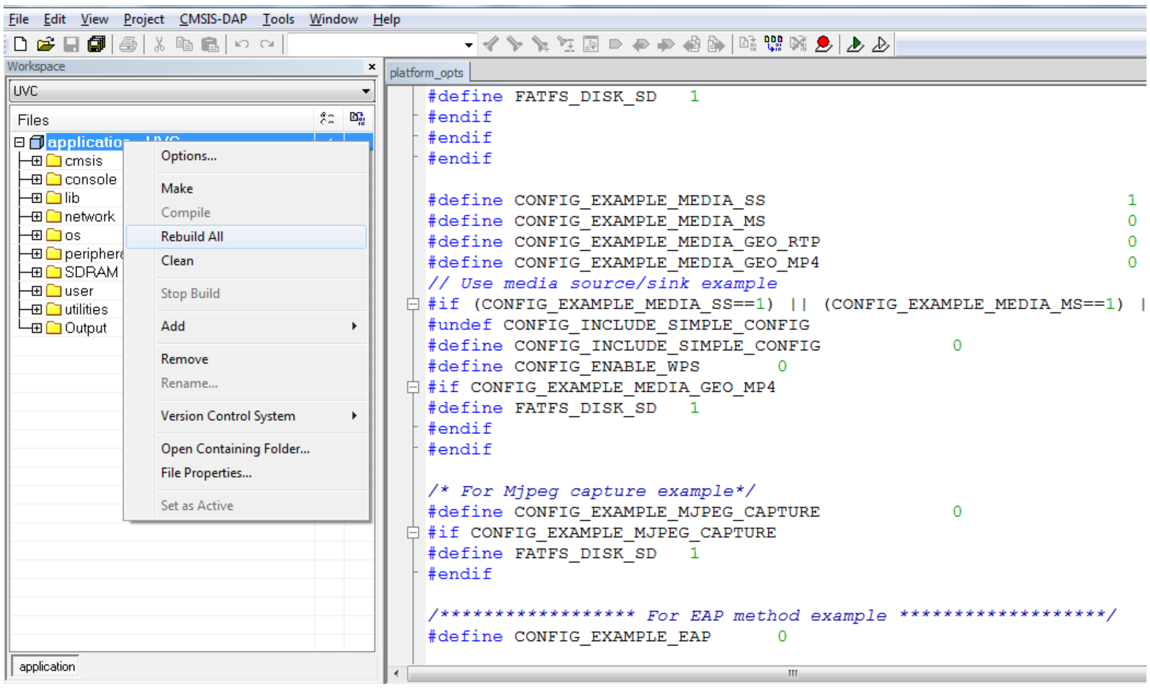


- After the UVC workspace is selected, the appropriate example, part of the workspace, needs to be enabled. Please open the platform_opts.h file in IAR IDE as shown below. Then, please set the "CONFIG_EXAMPLE_MEDIA_SS" to be "1"as shown in following figure and, thus, the media single stream example is enabled. The example code can be found in the project path, utilities→example→example_media_ss.c.



```
#define FATFS_DISK_SD    1
#endif
#endif
#endif

#define CONFIG_EXAMPLE_MEDIA_SS                                         1
#define CONFIG_EXAMPLE_MEDIA_MS                                         0
#define CONFIG_EXAMPLE_MEDIA_GEO_RTP                                    0
#define CONFIG_EXAMPLE_MEDIA_GEO_MP4                                    0
// Use media source/sink example
#if (CONFIG_EXAMPLE_MEDIA_SS==1) || (CONFIG_EXAMPLE_MEDIA_MS==1) ||
#undef CONFIG_INCLUDE_SIMPLE_CONFIG
#define CONFIG_INCLUDE_SIMPLE_CONFIG              0
#define CONFIG_ENABLE_WPS        0
#if CONFIG_EXAMPLE_MEDIA_GEO_MP4
#define FATFS_DISK_SD    1
#endif
#endif

/* For Mjpeg capture example*/
#define CONFIG_EXAMPLE_MJPEG_CAPTURE              0
#if CONFIG_EXAMPLE_MJPEG_CAPTURE
#define FATFS_DISK_SD    1
#endif

/****************** For EAP method example *******************/
#define CONFIG_EXAMPLE_EAP       0
```

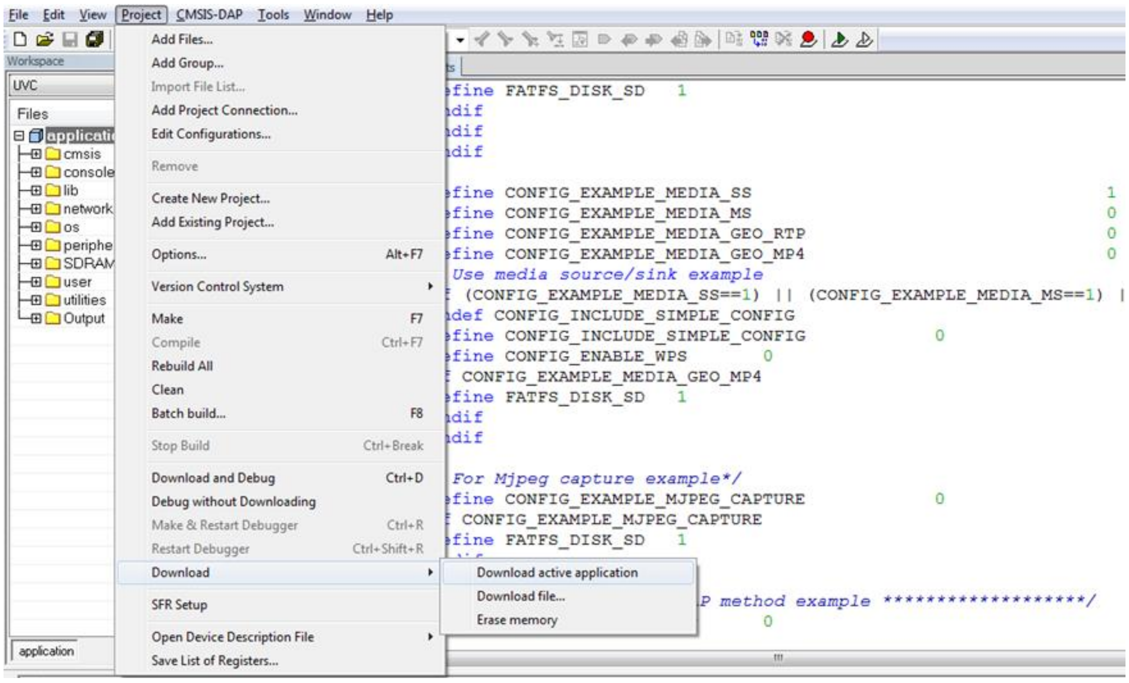
- Once all the configurations have been satisfied, please right click on project name, applications - UVC, in the left sidebar of IAR IDE as shown below and select "Rebuild All" option to build this project.



```
#define FATFS_DISK_SD    1
#endif
#endif
#endif

#define CONFIG_EXAMPLE_MEDIA_SS                                      1
#define CONFIG_EXAMPLE_MEDIA_MS                                      0
#define CONFIG_EXAMPLE_MEDIA_GEO_RTP                                 0
#define CONFIG_EXAMPLE_MEDIA_GEO_MP4                                 0
// Use media source/sink example
#if (CONFIG_EXAMPLE_MEDIA_SS==1) || (CONFIG_EXAMPLE_MEDIA_MS==1) |
#undef CONFIG_INCLUDE_SIMPLE_CONFIG
#define CONFIG_INCLUDE_SIMPLE_CONFIG            0
#define CONFIG_ENABLE_WPS          0
#if CONFIG_EXAMPLE_MEDIA_GEO_MP4
#define FATFS_DISK_SD    1
#endif
#endif

/* For Mjpeg capture example*/
#define CONFIG_EXAMPLE_MJPEG_CAPTURE            0
#if CONFIG_EXAMPLE_MJPEG_CAPTURE
#define FATFS_DISK_SD    1
#endif

/****************** For EAP method example ******************/
#define CONFIG_EXAMPLE_EAP       0
```

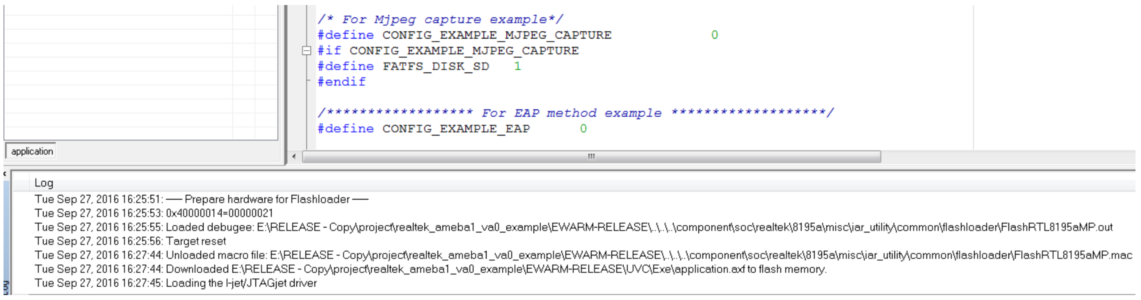
Flashing and Testing

- Before flashing the Ameba board, please set up a serial monitor via Tera Term or PuTTY and set up serial port to be COMX/38400, X: Port number. Once the build process is done and the Ameba board is connected, the flashing can be started by clicking the following in IAR IDE.
  Project→Download→Download Active Application



```
fine FATFS_DISK_SD    1
dif
dif
dif

fine CONFIG_EXAMPLE_MEDIA_SS                                      1
fine CONFIG_EXAMPLE_MEDIA_MS                                      0
fine CONFIG_EXAMPLE_MEDIA_GEO_RTP                                 0
fine CONFIG_EXAMPLE_MEDIA_GEO_MP4                                 0
Use media source/sink example
 (CONFIG_EXAMPLE_MEDIA_SS==1) || (CONFIG_EXAMPLE_MEDIA_MS==1) |
def CONFIG_INCLUDE_SIMPLE_CONFIG
fine CONFIG_INCLUDE_SIMPLE_CONFIG            0
fine CONFIG_ENABLE_WPS          0
 CONFIG_EXAMPLE_MEDIA_GEO_MP4
fine FATFS_DISK_SD    1
dif
dif

For Mjpeg capture example*/
fine CONFIG_EXAMPLE_MJPEG_CAPTURE            0
 CONFIG_EXAMPLE_MJPEG_CAPTURE
fine FATFS_DISK_SD    1

P method example ******************/
                  0
```

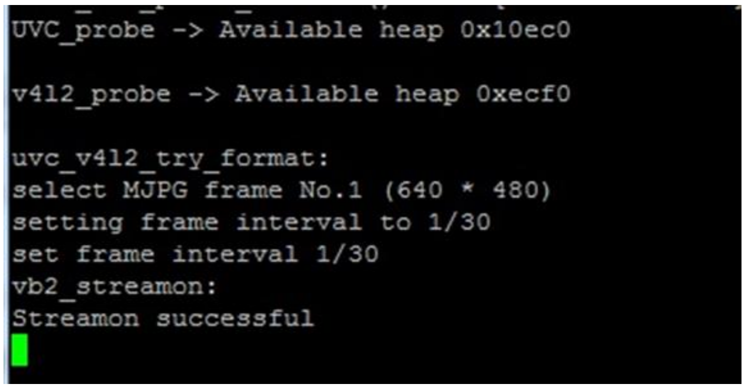
- The download and flashing process consumes more time. Please wait a moment. While the process is done, Please compare the messages in the Debug Log Window of the IAR project with that in the figure bellow. If the messages are similar or the same, the software is successfully flashed onto the Ameba board.



```
/* For Mjpeg capture example*/
#define CONFIG_EXAMPLE_MJPEG_CAPTURE            0
#if CONFIG_EXAMPLE_MJPEG_CAPTURE
#define FATFS_DISK_SD   1
#endif

/****************** For EAP method example ******************/
#define CONFIG_EXAMPLE_EAP       0
```

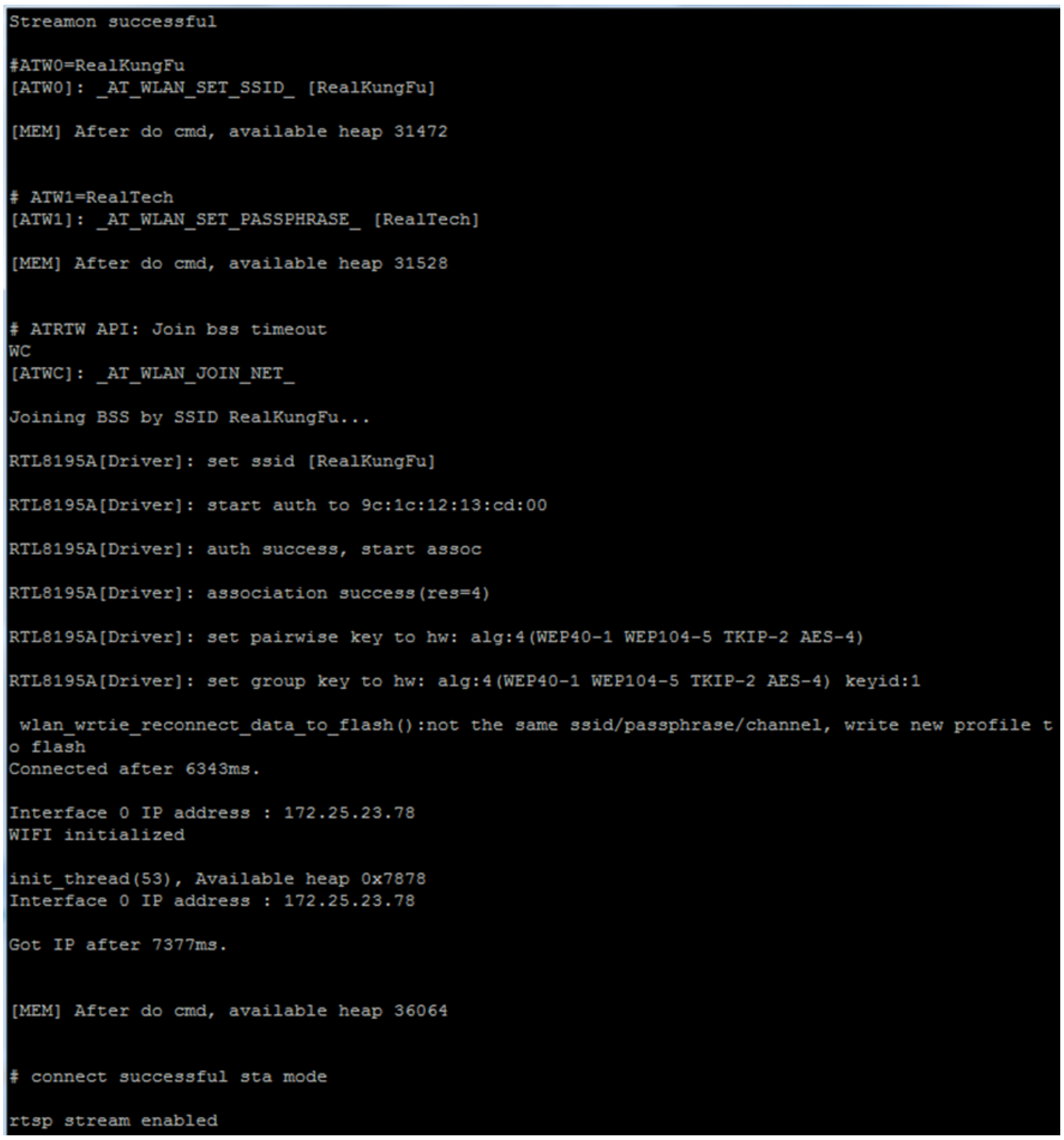
```
Tue Sep 27, 2016 16:25:51: —— Prepare hardware for Flashloader ——
Tue Sep 27, 2016 16:25:53: 0x40000014=00000021
Tue Sep 27, 2016 16:25:55: Loaded debugee: E:\RELEASE - Copy\project\realtek_ameba1_va0_example\EWARM-RELEASE\..\..\..\component\soc\realtek\8195a\misc\iar_utility\common\flashloader\FlashRTL8195aMP.out
Tue Sep 27, 2016 16:25:56: Target reset
Tue Sep 27, 2016 16:27:44: Unloaded macro file: E:\RELEASE - Copy\project\realtek_ameba1_va0_example\EWARM-RELEASE\..\..\..\component\soc\realtek\8195a\misc\iar_utility\common\flashloader\FlashRTL8195aMP.mac
Tue Sep 27, 2016 16:27:44: Downloaded E:\RELEASE - Copy\project\realtek_ameba1_va0_example\EWARM-RELEASE\UVC\Exe\application.axf to flash memory.
Tue Sep 27, 2016 16:27:45: Loading the I-jet/JTAGjet driver
```

- Next, please press the reset button on the Ameba board to restart the system. The log messages are shown as that in the following figure via Tera Term or PuTTY.



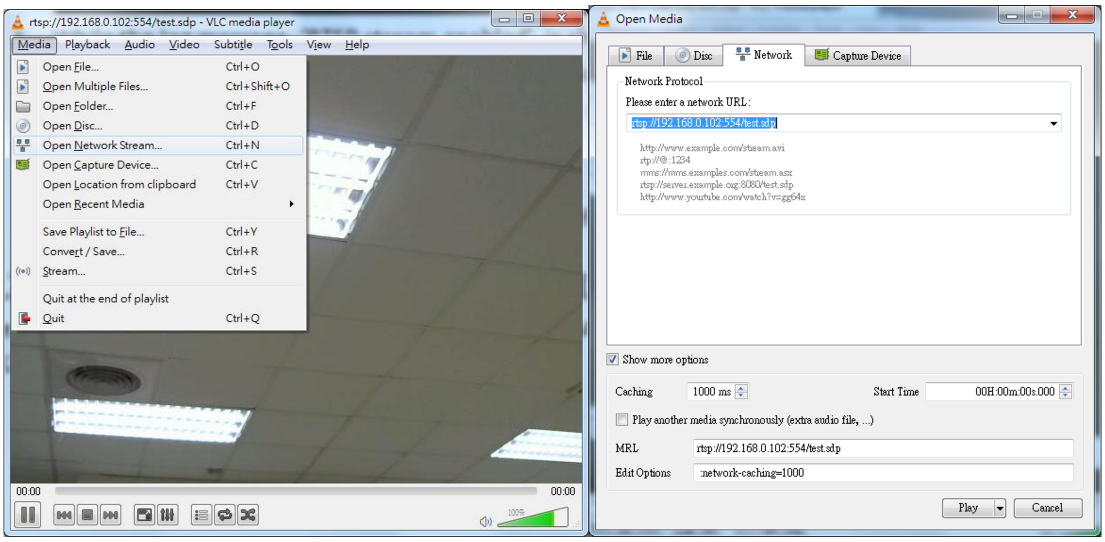
```
UVC_probe -> Available heap 0x10ec0

v4l2_probe -> Available heap 0xecf0

uvc_v4l2_try_format:
select MJPG frame No.1 (640 * 480)
setting frame interval to 1/30
set frame interval 1/30
vb2_streamon:
Streamon successful
```

- Next, it is going to set Ameba board to connect with an AP for RTSP streaming purpose. Please use the following AT commands to join a WiFi network via Tera Term or Putty. The following figure is an example for your reference.
  ATW0=<Name of WiFi SSID>: Set the WiFi AP to be connected
  ATW1=<Password>: Set the WiFi AP password
  ATWC: Initiate the connection



```
Streamon successful

#ATW0=RealKungFu
[ATW0]: _AT_WLAN_SET_SSID_ [RealKungFu]

[MEM] After do cmd, available heap 31472


# ATW1=RealTech
[ATW1]: _AT_WLAN_SET_PASSPHRASE_ [RealTech]

[MEM] After do cmd, available heap 31528


# ATRTW API: Join bss timeout
WC
[ATWC]: _AT_WLAN_JOIN_NET_

Joining BSS by SSID RealKungFu...

RTL8195A[Driver]: set ssid [RealKungFu]

RTL8195A[Driver]: start auth to 9c:1c:12:13:cd:00

RTL8195A[Driver]: auth success, start assoc

RTL8195A[Driver]: association success(res=4)

RTL8195A[Driver]: set pairwise key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4)

RTL8195A[Driver]: set group key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4) keyid:1

 wlan_wrtie_reconnect_data_to_flash():not the same ssid/passphrase/channel, write new profile t
o flash
Connected after 6343ms.

Interface 0 IP address : 172.25.23.78
WIFI initialized

init_thread(53), Available heap 0x7878
Interface 0 IP address : 172.25.23.78

Got IP after 7377ms.


[MEM] After do cmd, available heap 36064


# connect successful sta mode

rtsp stream enabled
```

- While the log message, “RTSP stream enabled”, is shown in the log console, please open the VLC media player, click “Media→Open Network Stream” option in menu bar, and type in: “rtsp://xxx.xxx.xxx.xxx:yyy/test.sdp” before clicking play button.
  xxx.xxx.xxx.xxx: the Ameba IP address.
  yyy: RTSP server port number (default is 554).
  Also, please select ‘Show more options’ checkbox if advanced setting is required.


rtsp://192.168.0.102:554/test.sdp - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Open File... Ctrl+O
Open Multiple Files... Ctrl+Shift+O
Open Folder... Ctrl+F
Open Disc... Ctrl+D
Open Network Stream... Ctrl+N
Open Capture Device... Ctrl+C
Open Location from clipboard Ctrl+V
Open Recent Media
Save Playlist to File... Ctrl+Y
Convert / Save... Ctrl+R
Stream... Ctrl+S
Quit at the end of playlist
Quit Ctrl+Q
00:00
00:00
Open Media
File
Disc
Network
Capture Device
Play
Cancel

Network Protocol

Please enter a network URL: rtsp://192.168.0.102:554/test.sdp

http://www.example.com/stream.avi
rtp://@:1234
mms://mms.examples.com/stream.asx
rtsp://server.example.org:8080/test.sdp
http://www.youtube.com/watch?v=gg64x

☑ Show more options

Caching 1000 ms Start Time 00H:00m:00s.000

☐ Play another media synchronously (extra audio file, ...)

MRL rtsp://192.168.0.102:554/test.sdp

Edit Options :network-caching=1000

2.6 Media Single Stream – Source I2S, Sink RTSP Stream

Please refer to following steps to understand how to run the MMF example with source as ALC5651 in I2S codec and sink as RTSP stream.

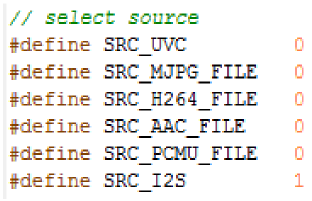
Pre-requisites

- Ameba board
- ALC5651 Ext. board
- Audio connection cable (3.5mm)
- USB cable is used to connect with PC.
- WiFi connection for sending RTSP streaming out
- VLC media player on the testing PC

Procedures

- Please set “CONFIG_EXAMPLE_MEDIA_SS” to be “1” in “platform_opts.h”. The example code can be found in the project path, utilities→example→example_media_ss.c.

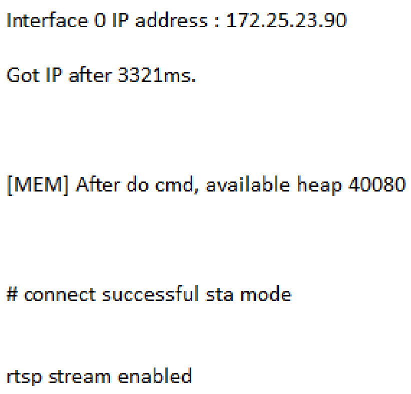
- Please set “SRC_I2S” to be “1” in “example_media_ss.c” and other source definitions are set to be “0” as shown in the following figure.



```
// select source
#define SRC_UVC          0
#define SRC_MJPG_FILE    0
#define SRC_H264_FILE    0
#define SRC_AAC_FILE     0
#define SRC_PCMU_FILE    0
#define SRC_I2S          1
```

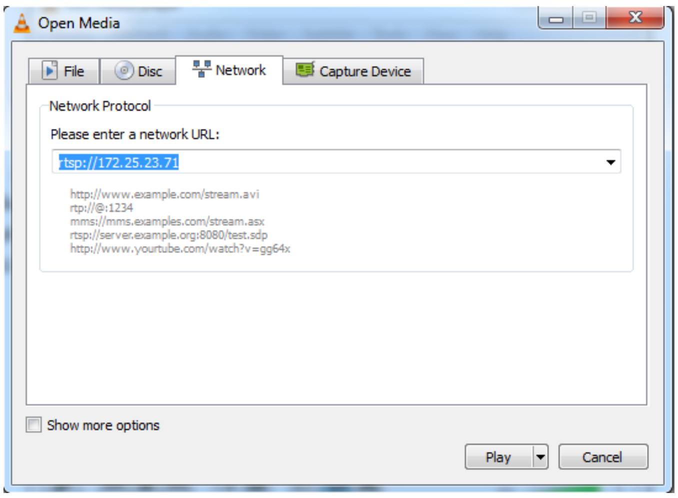
In addition, the source code of SRC_I2S is in mmf_source_i2s_file.[c/h](component\common\media\framework\mmf_source_modu les)

- Please compile the project and download the application into Ameba board as described in the previous section.
- Next, please connect the ALC5651 Ext. board with Ameba board. Then, please also connect the audio connection cable to line-in, labelled as “PH3”, on ALC5651 Ext. board.
- As described in the previous section, please connect both the testing PC and Ameba to the same WiFi AP. As a result, the following figure can be shown. While the log message, "rtsp stream enabled", is shown in the log console, it means that the RTSP server is ready.



```
Interface 0 IP address : 172.25.23.90

Got IP after 3321ms.


[MEM] After do cmd, available heap 40080


# connect successful sta mode


rtsp stream enabled
```

- Next, please open the VLC media player, click “Media→Open Network Stream” option in menu bar, and type in: “rtsp://xxx.xxx.xxx.xxx” before clicking play button. xxx.xxx.xxx.xxx: the Ameba IP address.
  Also, please select ‘Show more options’ checkbox if advanced setting is required.



Please enter a network URL:

```
rtsp://172.25.23.71
```

```
http://www.example.com/stream.avi
rtp://@:1234
mms://mms.examples.com/stream.asx
rtsp://server.example.org:8080/test.sdp
http://www.yourtube.com/watch?v=gg64x
```

2.7 Media Single Stream – Source UVC Camera, Sink SD Card

Please refer to following steps to understand how to run the MMF example with source as UVC camera and sink as SD card.

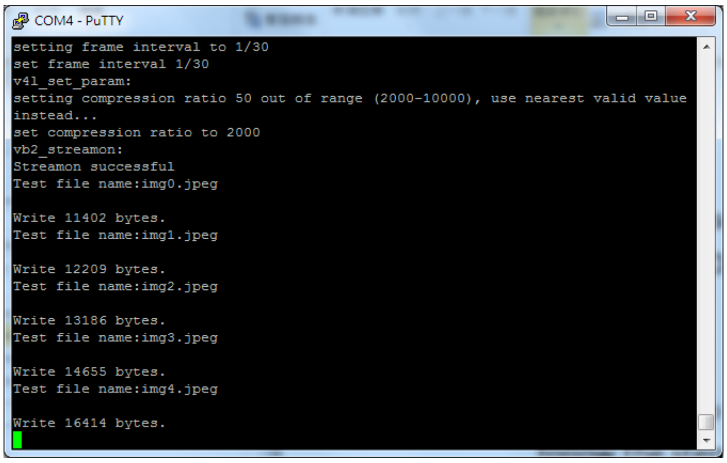
Pre-requisites

- Ameba board

- UVC camera
- USB cable is used to connect with PC.
- SD card and SD/MMC card connector

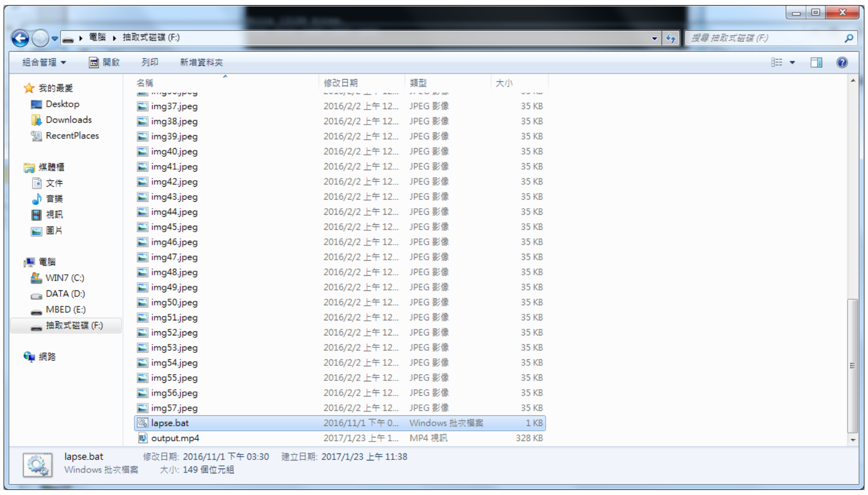
Procedures

- Please refer to the document (UM0073 Realtek Ameba-1 Storage User Manual.pdf) to follow the steps in Section 2 to connect SD/MMC card connector with Ameba board. Then, please also plug SD card into SD/MMC card connector.
- Please set “CONFIG_EXAMPLE_TIMELAPSE” to be “1” in “platform_opts.h”. The example code can be found in the project path, utilities→example→example_media_tl.c.
- Please compile the project and download the application into Ameba board as described in the previous section.
- While the log message, "Streamon successful", is shown in the log console as the figure bellow, it is going to write jpeg files into SD card.



```
setting frame interval to 1/30
set frame interval 1/30
v4l_set_param:
setting compression ratio 50 out of range (2000-10000), use nearest valid value
instead...
set compression ratio to 2000
vb2_streamon:
Streamon successful
Test file name:img0.jpeg

Write 11402 bytes.
Test file name:img1.jpeg

Write 12209 bytes.
Test file name:img2.jpeg

Write 13186 bytes.
Test file name:img3.jpeg

Write 14655 bytes.
Test file name:img4.jpeg

Write 16414 bytes.
```

- Please wait a while then pull out the SD card from the SD card connector on the Ameba board. Then, please also connect SD card with another PC or laptop to check whether the jpeg files exist or not, as shown in the figure bellow. Then, please download “ffmpeg.exe” from FFmpeg website and execute the batch file, “lapse.bat”, in the path (component\common\example\media_time_lapse) to generate a time-lapse mp4 file.



名稱	修改日期	類型	大小
img37.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img38.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img39.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img40.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img41.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img42.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img43.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img44.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img45.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img46.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img47.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img48.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img49.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img50.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img51.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img52.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img53.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img54.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img55.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img56.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
img57.jpeg	2016/2/2 上午 12...	JPEG 影像	35 KB
lapse.bat	2016/11/1 下午 0...	Windows 批次檔案	1 KB
output.mp4	2017/1/23 上午 1...	MP4 視訊	328 KB

lapse.bat 修改日期: 2016/11/1 下午 03:30 建立日期: 2017/1/23 上午 11:38
Windows 批次檔案 大小: 149 個位元組

2.8 Media Single Stream – Source RTP Stream, Sink I2S

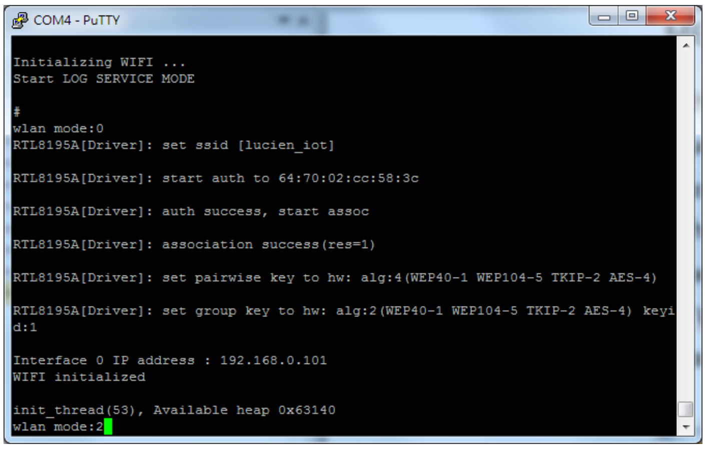
Please refer to following steps to understand how to run the MMF example with source as RTP audio stream and sink as I2S.

Pre-requisites

- Ameba board
- ALC5651 Ext. board
- Earphone
- USB cable is used to connect with PC.
- WiFi connection for receiving RTP streaming
- VLC media player on the testing PC
- ffmpeg on the testing PC

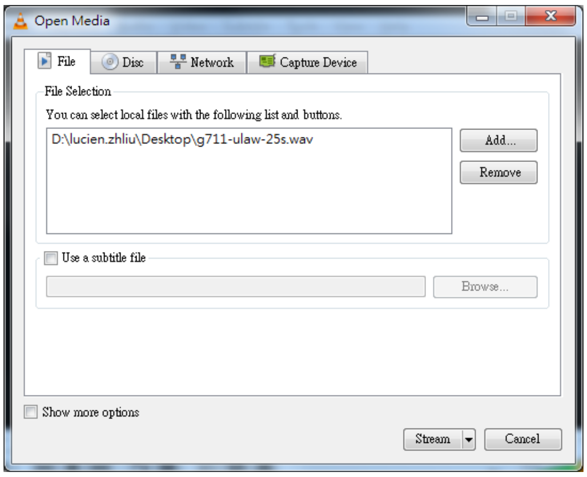
Procedures

- Please set “CONFIG_EXAMPLE_MEDIA_AUDIO_FROM_RTP” to be “1” in “platform_opts.h”. The example code can be found in the project path, utilities→example→ example_media_audio_from_rtp.c.
- In order to achieve the better audio quality but not required, please try to modify the following definitions for improving the WLAN performance for RTP packets. The following is an example.
  In lwipopts.h,
  ```
  PBUF_POOL_SIZE 20 => 40
  PBUF_POOL_BUFSIZE 500 => 250
  DEFAULT_UDP_RECVMBOX_SIZE 6 => 24
  ```
  In opt.h,
  ```
  MEMP_NUM_NETBUF 2 => 24
  ```
- Please compile the project and download the application into Ameba board as described in the previous section.
- Next, please connect the ALC5651 Ext. board with Ameba board. Then, please also connect the earphone to audio-out, labelled as “PH2”, on ALC5651 Ext. board.
- As described in the previous section, please connect both the testing PC and Ameba to the same WiFi AP. As a result, the following figure can be shown. While the log message, "wlan mode:2", is shown in the log console, it means that the RTP server is ready.


COM4 - PuTTY

```
Initializing WIFI ...
Start LOG SERVICE MODE

#
wlan mode:0
RTL8195A[Driver]: set ssid [lucien_iot]

RTL8195A[Driver]: start auth to 64:70:02:cc:58:3c

RTL8195A[Driver]: auth success, start assoc

RTL8195A[Driver]: association success(res=1)

RTL8195A[Driver]: set pairwise key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4)

RTL8195A[Driver]: set group key to hw: alg:2(WEP40-1 WEP104-5 TKIP-2 AES-4) keyi
d:1

Interface 0 IP address : 192.168.0.101
WIFI initialized

init_thread(53), Available heap 0x63140
wlan mode:2
```

- Next, please open the VLC media player, click "Media→Stream…" option in menu bar, and add the WAV file (G711 only) as the bellow figure. Then, click "Stream".



File Selection

You can select local files with the following list and buttons.

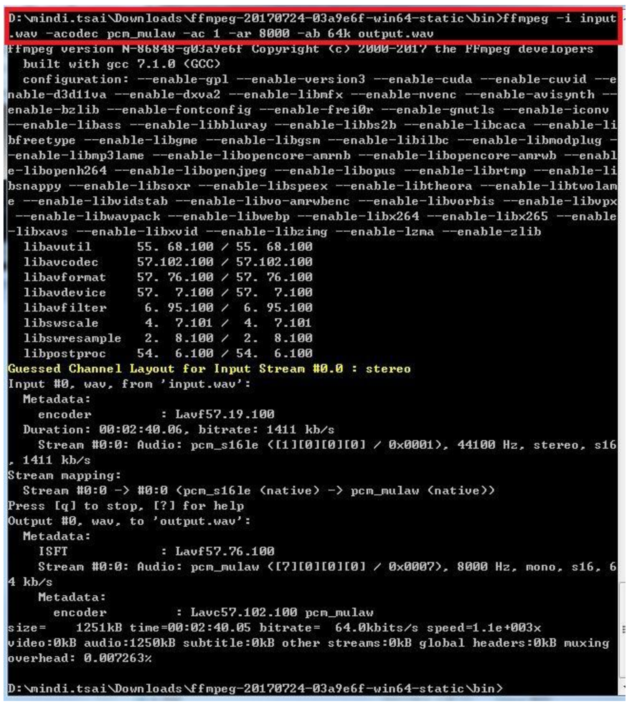
D:\lucien.zhliu\Desktop\g711-ulaw-25s.wav

☐ Use a subtitle file

☐ Show more options

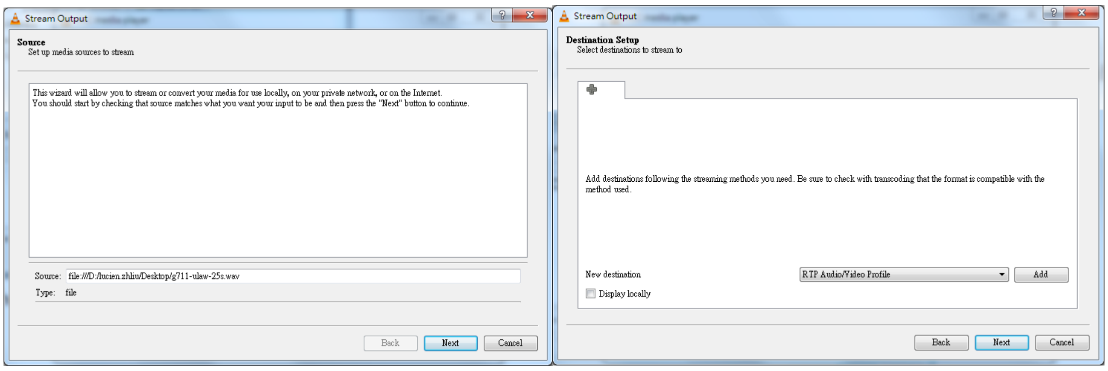
NOTE: Use ffmpeg to generate (transcode) compatible WAV file.

ffmpeg -i **input.wav** -acodec pcm_mulaw -ac 1 -ar 8000 -ab 64k **output.wav**



```
D:\mindi.tsai\Downloads\ffmpeg-20170724-03a9e6f-win64-static\bin>ffmpeg -i input
.wav -acodec pcm_mulaw -ac 1 -ar 8000 -ab 64k output.wav
ffmpeg version N-86848-g03a9e6f Copyright (c) 2000-2017 the FFmpeg developers
  built with gcc 7.1.0 (GCC)
  configuration: --enable-gpl --enable-version3 --enable-cuda --enable-cuvid --e
nable-d3d11va --enable-dxva2 --enable-libmfx --enable-nvenc --enable-avisynth --
enable-bzlib --enable-fontconfig --enable-frei0r --enable-gnutls --enable-iconv
--enable-libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-li
bfreetype --enable-libgme --enable-libgsm --enable-libilbc --enable-libmodplug -
-enable-libmp3lame --enable-libopencore-amrnb --enable-libopencore-amrwb --enabl
e-libopenh264 --enable-libopenjpeg --enable-libopus --enable-librtmp --enable-li
bsnappy --enable-libsoxr --enable-libspeex --enable-libtheora --enable-libtwolam
e --enable-libvidstab --enable-libvo-amrwbenc --enable-libvorbis --enable-libvpx
 --enable-libwavpack --enable-libwebp --enable-libx264 --enable-libx265 --enable
-libxavs --enable-libxvid --enable-libzimg --enable-lzma --enable-zlib
  libavutil      55. 68.100 / 55. 68.100
  libavcodec     57.102.100 / 57.102.100
  libavformat    57. 76.100 / 57. 76.100
  libavdevice    57.  7.100 / 57.  7.100
  libavfilter     6. 95.100 /  6. 95.100
  libswscale      4.  7.101 /  4.  7.101
  libswresample   2.  8.100 /  2.  8.100
  libpostproc    54.  6.100 / 54.  6.100
Guessed Channel Layout for Input Stream #0.0 : stereo
Input #0, wav, from 'input.wav':
  Metadata:
    encoder         : Lavf57.19.100
  Duration: 00:02:40.06, bitrate: 1411 kb/s
    Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, stereo, s16
, 1411 kb/s
Stream mapping:
  Stream #0:0 -> #0:0 (pcm_s16le (native) -> pcm_mulaw (native))
Press [q] to stop, [?] for help
Output #0, wav, to 'output.wav':
  Metadata:
    ISFT            : Lavf57.76.100
    Stream #0:0: Audio: pcm_mulaw ([7][0][0][0] / 0x0007), 8000 Hz, mono, s16, 6
4 kb/s
    Metadata:
      encoder         : Lavc57.102.100 pcm_mulaw
size=    1251kB time=00:02:40.05 bitrate=  64.0kbits/s speed=1.1e+003x
video:0kB audio:1250kB subtitle:0kB other streams:0kB global headers:0kB muxing
overhead: 0.007263%

D:\mindi.tsai\Downloads\ffmpeg-20170724-03a9e6f-win64-static\bin>
```

- In the window as the following figure, please click “Next” and, in the next window, set the “New destination” to be “RTP Audio/Video Profile” and click “Add”.


Stream Output

Source
Set up media sources to stream

This wizard will allow you to stream or convert your media for use locally, on your private network, or on the Internet.
You should start by checking that source matches what you want your input to be and then press the "Next" button to continue.

Source: file:///D:/lucien.zhliu/Desktop/g711-ulaw-25s.wav
Type: file

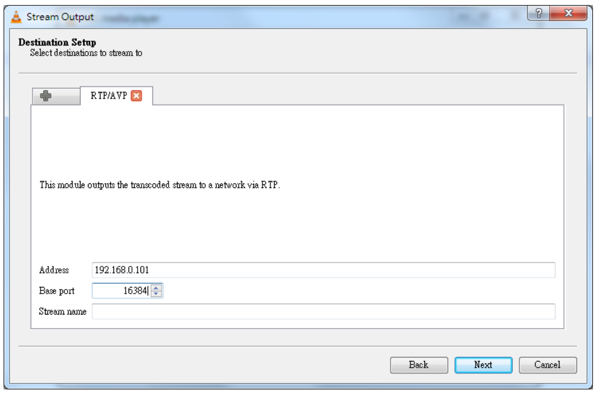
Stream Output

Destination Setup
Select destinations to stream to

Add destinations following the streaming methods you need. Be sure to check with transcoding that the format is compatible with the method used.

New destination: RTP Audio/Video Profile
Display locally

- Please refer to the following figure and type the Ameba board’s IP address and the RTP port (the default value is 16384). Then, click “Next”.


Stream Output

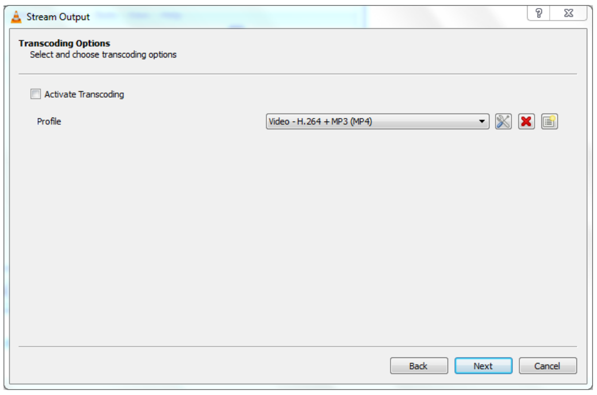
Destination Setup
Select destinations to stream to

RTP/AVP

This module outputs the transcoded stream to a network via RTP.

Address: 192.168.0.101
Base port: 16384
Stream name:

- In this step, please uncheck “Active Transcoding” as shown in the following figure and click “Next”.

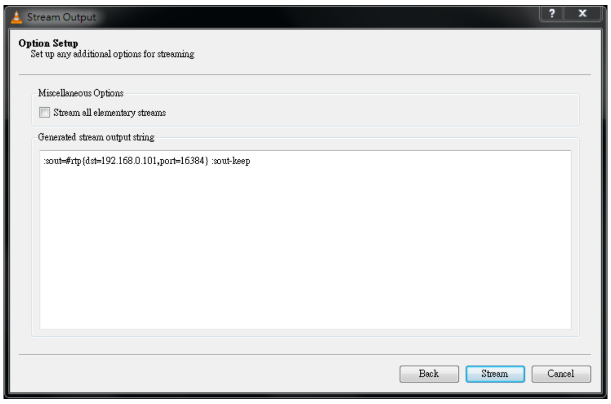

Stream Output

Transcoding Options
Select and choose transcoding options

Activate Transcoding

Profile: Video - H.264 + MP3 (MP4)

- Finally, the VLC generated stream output string is the same as that in the bellow figure, and click “Stream” to start the RTP transmission. Please also check the earphone plugged in the ALC5651 Ext. board whether the voice is out or not.



Option Setup
Set up any additional options for streaming

Miscellaneous Options
☐ Stream all elementary streams

Generated stream output string

```
:sout=#rtp{dst=192.168.0.101,port=16384} :sout-keep
```

2.9 Media Multiple Streams – Source GEO Camera, Sink RTP Stream

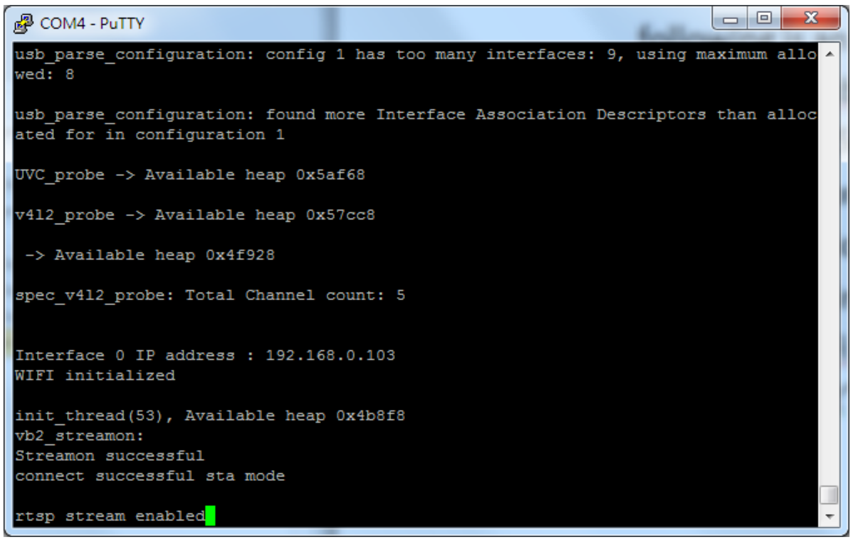
Please refer to following steps to understand how to run the MMF example with source as GEO camera and sink as RTSP stream.

<u>Pre-requisites</u>

- Ameba board
- GEO camera
- USB cable is used to connect with PC.
- WiFi connection for sending RTSP streaming out
- USB OTG cable is used to connect GEO camera to Ameba board.

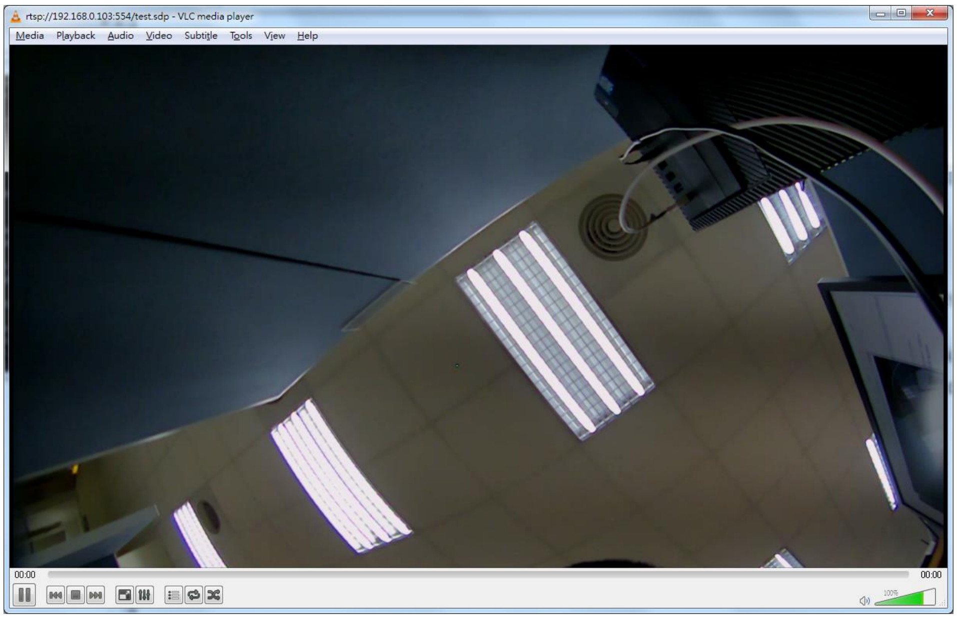
<u>Procedures</u>

- As the Hardware Setup is presented in Section 3.1, please connect GEO camera, instead of UVC camera, to the Ameba board.
- Please set “`CONFIG_EXAMPLE_MEDIA_MS`” to be “1” in “platform_opts.h”. The example code can be found in the project path, utilities→example→ example_media_ms.c.
- In order to achieve the better streaming quality but not required, please try to modify the following definition for improving the WLAN performance for RTSP packets. The following is an example.
  In rtw_opt_skbuf.c,
  MAX_SKB_BUF_NUM 8 => 12
- Please compile the project and download the application into Ameba board as described in the previous section.
- As described in the previous section, please connect both the testing PC and Ameba to the same WiFi AP. As a result, the following figure can be shown. While the log message, "Streamon successful", is shown in the log console, it means that the RTSP server is ready.



```
usb_parse_configuration: config 1 has too many interfaces: 9, using maximum allo
wed: 8

usb_parse_configuration: found more Interface Association Descriptors than alloc
ated for in configuration 1

UVC_probe -> Available heap 0x5af68

v4l2_probe -> Available heap 0x57cc8

 -> Available heap 0x4f928

spec_v4l2_probe: Total Channel count: 5


Interface 0 IP address : 192.168.0.103
WIFI initialized

init_thread(53), Available heap 0x4b8f8
vb2_streamon:
Streamon successful
connect successful sta mode

rtsp stream enabled
```

- Next, please open the VLC media player, click "Media→Open Network Stream" option in menu bar, and type in: "rtsp://xxx.xxx.xxx.xxx:yyy/test.sdp" before clicking play button.
  xxx.xxx.xxx.xxx: the Ameba IP address.
  yyy: RTSP server port number (default is 554).
- Please check whether both video and audio work normally or not by the VLC media player, in the testing PC, as shown in the figure as bellow.



2.10Media Multiple Streams – Source GEO Camera, Sink SD Card

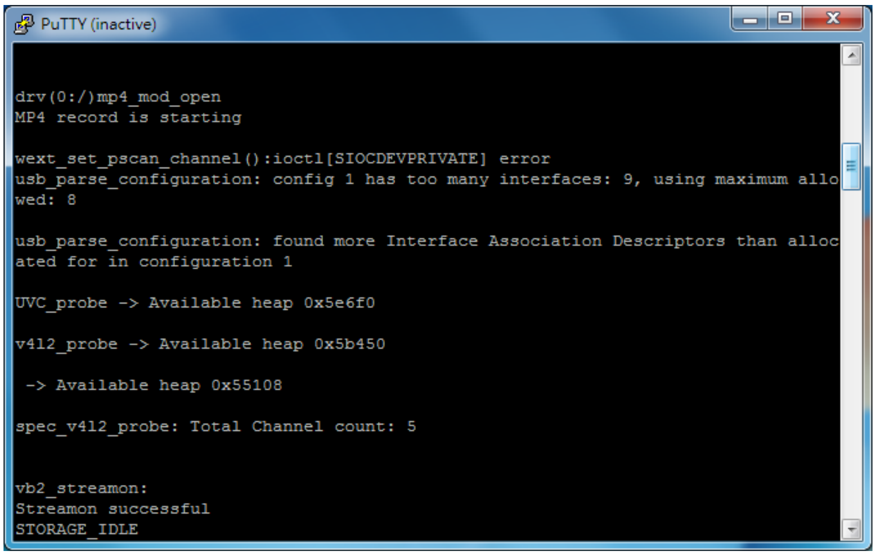
Please refer to following steps to understand how to run the MMF example with source as GEO camera and sink as SD card.

Pre-requisites

- Ameba board
- GEO camera
- USB cable is used to connect with PC.
- SD card and SD/MMC card connector

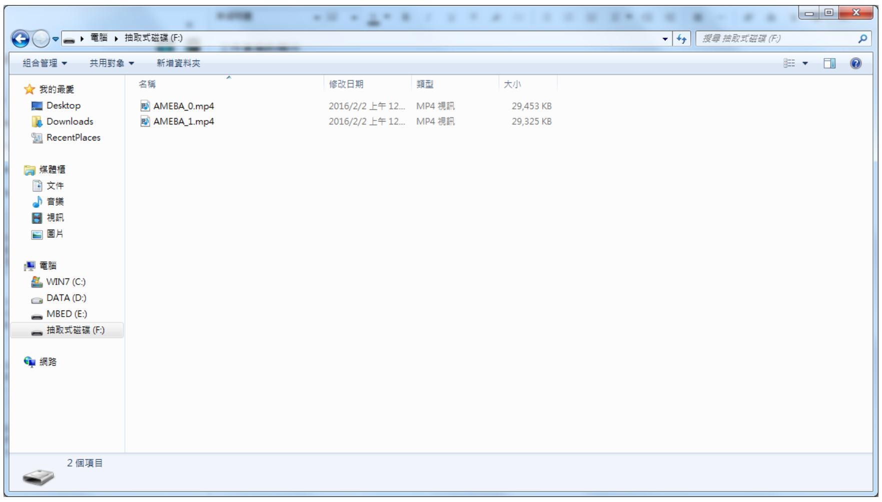
Procedures

- Please refer to the document (UM0073 Realtek Ameba-1 Storage User Manual.pdf) to follow the steps in Section 2 to connect SD/MMC card connector with Ameba board. Then, please also plug SD card into SD/MMC card connector.
- Please set "CONFIG_EXAMPLE_MEDIA_GEO_MP4" to be "1" in "platform_opts.h". The example code can be found in the project path, utilities→example→ example_media_geo_mp4.c.
- Please compile the project and download the application into Ameba board as described in the previous section.
- While the log message, "Streamon successful", is shown in the log console as the figure bellow, it is going to write mp4 file into SD card.



```
drv(0:/)mp4_mod_open
MP4 record is starting

wext_set_pscan_channel():ioctl[SIOCDEVPRIVATE] error
usb_parse_configuration: config 1 has too many interfaces: 9, using maximum allo
wed: 8

usb_parse_configuration: found more Interface Association Descriptors than alloc
ated for in configuration 1

UVC_probe -> Available heap 0x5e6f0

v4l2_probe -> Available heap 0x5b450

 -> Available heap 0x55108

spec_v4l2_probe: Total Channel count: 5


vb2_streamon:
Streamon successful
STORAGE_IDLE
```

- Please wait a while then pull out the SD card from the SD card connector on the Ameba board. Then, please also connect SD card with another PC or laptop to check whether the mp4 files exist or not, as shown in the figure bellow, and whether both audio and video work normally by the VLC media player or not.



名稱	修改日期	類型	大小
AMEBA_0.mp4	2016/2/2 上午 12...	MP4 視訊	29,453 KB
AMEBA_1.mp4	2016/2/2 上午 12...	MP4 視訊	29,325 KB

2.11MP3 Media Streaming Over RTP (Source-RTP; Sink Mp3 I2S).

This example highlights the streaming of mp3 files over wifi to the ameba and decoding of the mp3 locally in the ameba board and audio output from the audio jack. The example has two methods to provide audio output.

- Audio output using the ALC5651 board.
- Audio output using SGTL5000 board.

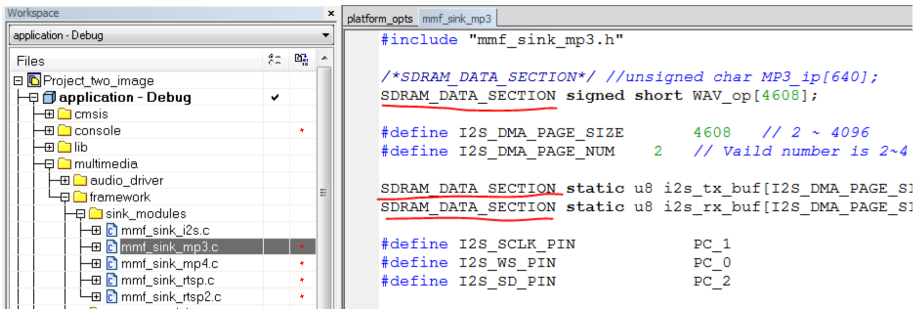
Pre-requisites

- Ameba board
- USB cable is used to connect with PC.
- ALC5651 Extension board or SGTL5000 Extension board.
- 3.5mm Audio jack to hear audio output.

Procedures

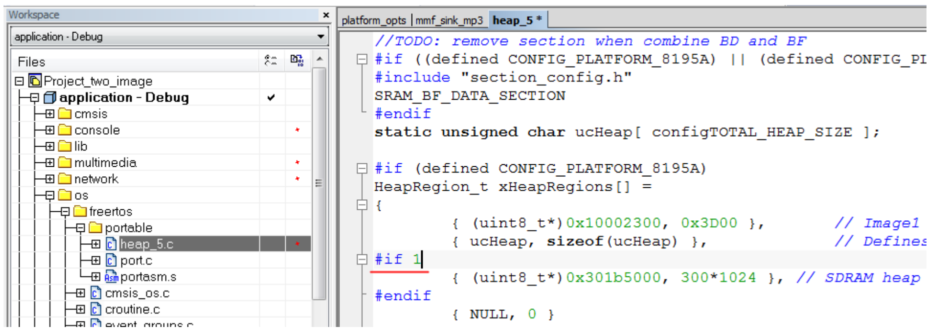
- In platform_opts.h edit the macro “CONFIG_EXAMPLE_MP3_STREAM_RTP” and set it to 1.
- By default the example uses the ALC5651 extension board is used for audio output but in case you wish to use the SGTL5000 audio extension board please set the macro “CONFIG_EXAMPLE_MP3_STREAM_SGTL5000” to one as well in the platform_opts.h file.
- The ALC5651 extension board fits like a shield on the AMEBA but in order to setup the SGTL5000 board please refer to :- http://www.amebaiot.com/en/standard-sdk-i2s-audio-demo/

- In order to ensure that the application fits within the memory constraints , please ensure that the respective sink module “mmf_sink_mp3.c” is modified as shown below.



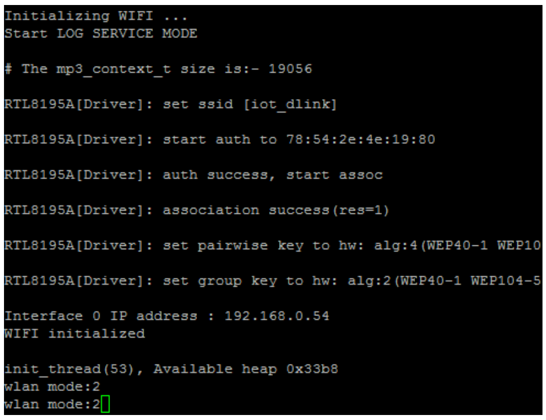
```
#include "mmf_sink_mp3.h"

/*SDRAM_DATA_SECTION*/ //unsigned char MP3_ip[640];
SDRAM_DATA_SECTION signed short WAV_op[4608];

#define I2S_DMA_PAGE_SIZE       4608   // 2 ~ 4096
#define I2S_DMA_PAGE_NUM     2    // Vaild number is 2~4

SDRAM_DATA_SECTION static u8 i2s_tx_buf[I2S_DMA_PAGE_SI
SDRAM_DATA_SECTION static u8 i2s_rx_buf[I2S_DMA_PAGE_SI

#define I2S_SCLK_PIN            PC_1
#define I2S_WS_PIN              PC_0
#define I2S_SD_PIN              PC_2
```

- Also to ensure that there is enough heap please enable the SDRAM heap as shown below in case of AMEBA1.



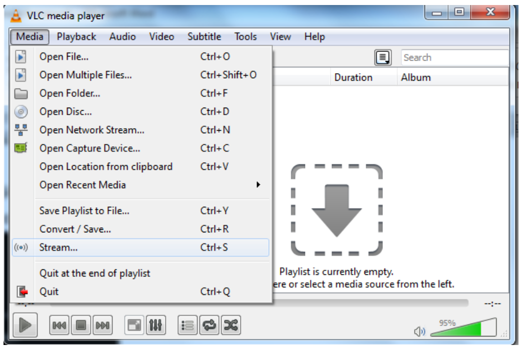
```
//TODO: remove section when combine BD and BF
#if ((defined CONFIG_PLATFORM_8195A) || (defined CONFIG_PI
#include "section_config.h"
SRAM_BF_DATA_SECTION
#endif
static unsigned char ucHeap[ configTOTAL_HEAP_SIZE ];

#if (defined CONFIG_PLATFORM_8195A)
HeapRegion_t xHeapRegions[] =
{
        { (uint8_t*)0x10002300, 0x3D00 },        // Image1
        { ucHeap, sizeof(ucHeap) },             // Defines
#if 1
        { (uint8_t*)0x301b5000, 300*1024 }, // SDRAM heap
#endif
        { NULL, 0 }
```

- Once the macros are set please re-build the application project and flash the AMEBA board appropriately.
- Once flashed reset the AMEBA board and connect the board to wi-fi using the AT Commands and ensure the board gets an ip address as shown below.

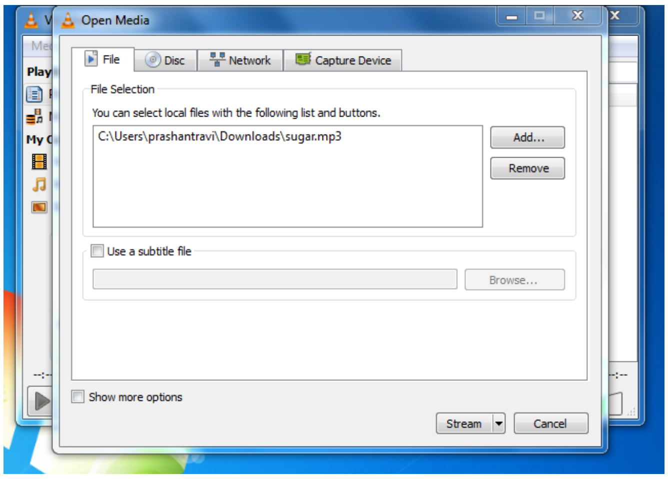


```
Initializing WIFI ...
Start LOG SERVICE MODE

# The mp3_context_t size is:- 19056

RTL8195A[Driver]: set ssid [iot_dlink]

RTL8195A[Driver]: start auth to 78:54:2e:4e:19:80

RTL8195A[Driver]: auth success, start assoc

RTL8195A[Driver]: association success(res=1)

RTL8195A[Driver]: set pairwise key to hw: alg:4(WEP40-1 WEP10

RTL8195A[Driver]: set group key to hw: alg:2(WEP40-1 WEP104-5

Interface 0 IP address : 192.168.0.54
WIFI initialized

init_thread(53), Available heap 0x33b8
wlan mode:2
wlan mode:2
```

- Once this is done, Open vlc media player on any other device connected to the same wifi network. Select the stream option(media->stream) as shown below.


VLC media player
Media
Playback
Audio
Video
Subtitle
Tools
View
Help
Open File... Ctrl+O
Open Multiple Files... Ctrl+Shift+O
Open Folder... Ctrl+F
Open Disc... Ctrl+D
Open Network Stream... Ctrl+N
Open Capture Device... Ctrl+C
Open Location from clipboard Ctrl+V
Open Recent Media
Save Playlist to File... Ctrl+Y
Convert / Save... Ctrl+R
Stream... Ctrl+S
Quit at the end of playlist
Quit Ctrl+Q
Search
Duration
Album
Playlist is currently empty.
ere or select a media source from the left.
95%

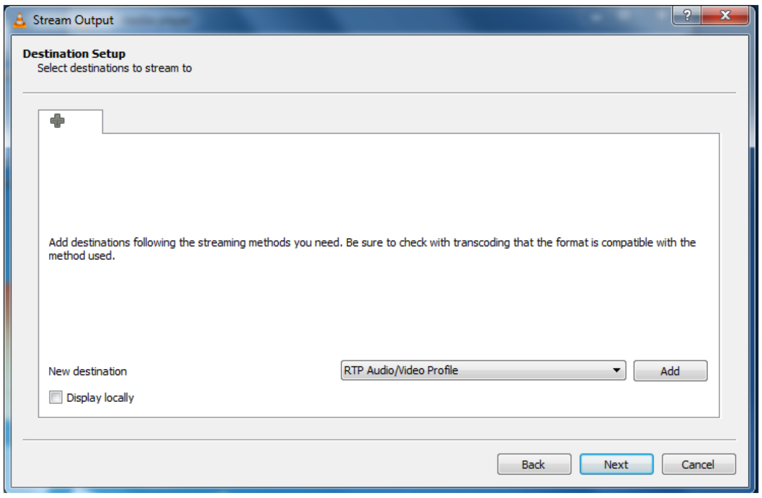
- Once you select stream, use add and select the mp3 file that you wish to stream over rtp as shown below.



You can select local files with the following list and buttons.

C:\Users\prashantravi\Downloads\sugar.mp3

- Once you select the mp3 file click the stream option and proceed to the next window. Click next again and till you reach the "Destination Support" tab and select the destination as "RTP Audio/Video Profile" as shown below and click "Add"

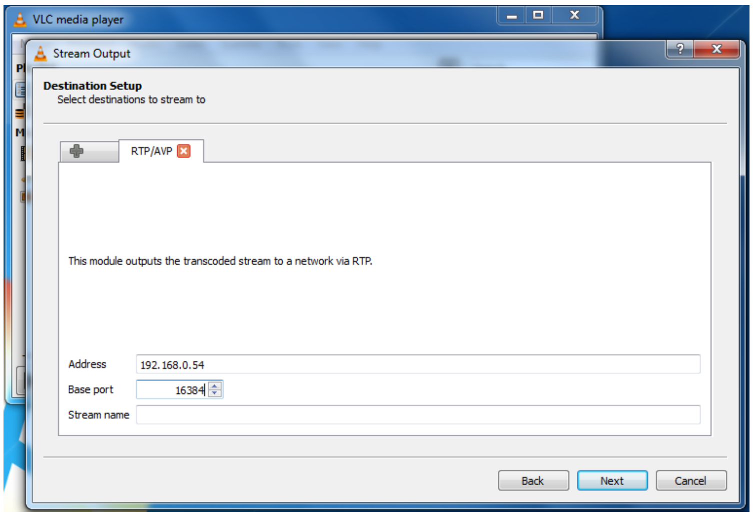


Destination Setup
Select destinations to stream to

Add destinations following the streaming methods you need. Be sure to check with transcoding that the format is compatible with the method used.

New destination: RTP Audio/Video Profile

- Once in add, please enter the ip address of the ameba board that was assigned when you connected to the wifi hotspot. Enter the base port address as: - 16384 as shown below.



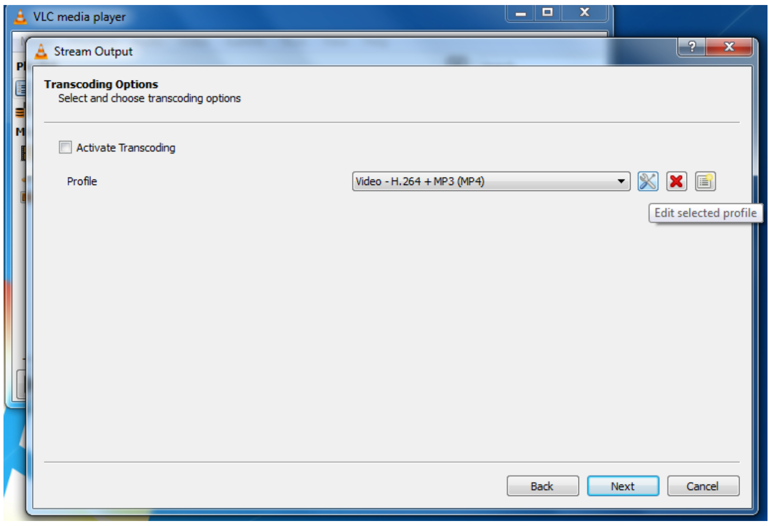
Destination Setup
Select destinations to stream to

RTP/AVP

This module outputs the transcoded stream to a network via RTP.

Address: 192.168.0.54
Base port: 16384
Stream name:

- Click next and select the streaming option “Video – H.264+MP3(MP4)” after this click the settings button just beside the dropdown menu as shown below.

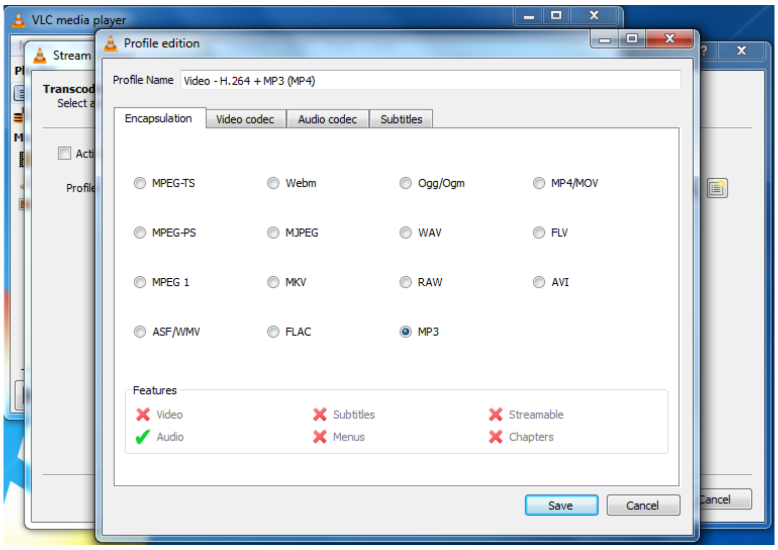


Transcoding Options
Select and choose transcoding options

☐ Activate Transcoding

Profile: Video - H.264 + MP3 (MP4)

- Once in the settings option MP3 from the checkboxes as shown below.

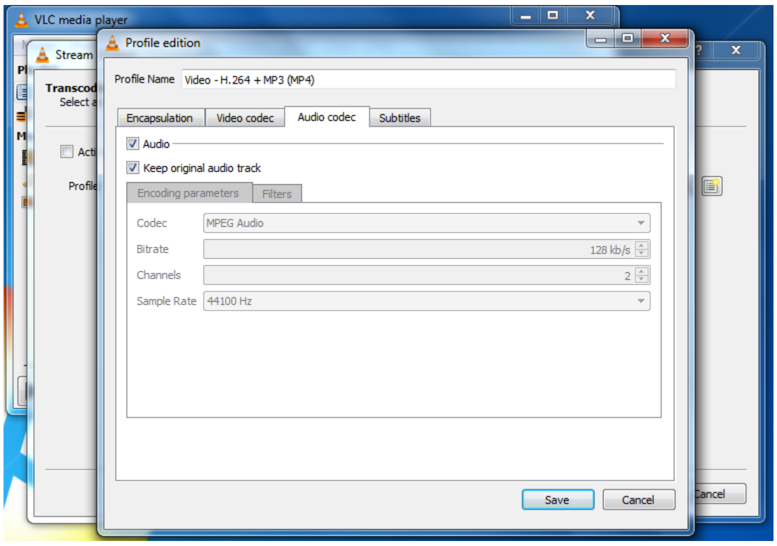


Profile Name: Video - H.264 + MP3 (MP4)

MPEG-TS, Webm, Ogg/Ogm, MP4/MOV, MPEG-PS, MJPEG, WAV, FLV, MPEG 1, MKV, RAW, AVI, ASF/WMV, FLAC, [✓ MP3]

Features
✗ Video
✓ Audio
✗ Subtitles
✗ Menus
✗ Streamable
✗ Chapters

- Once the mp3 option is selected, go to the “Audio Codec” tab and select the checkbox “Keep original audio track” as shown below.



Profile Name: Video - H.264 + MP3 (MP4)

☑ Audio

☑ Keep original audio track

Codec: MPEG Audio

Bitrate: 128 kb/s

Channels: 2

Sample Rate: 44100 Hz

- Once the audio codec is set, click save, then click next and click "stream"
- Once the streaming begins in vlc media player, connect the audio jack to the AMEBA in order to hear the audio output in the earphones. Both AMEBA and the device where vlc media player is streaming should be connected to the same Wi-Fi network.
- To ensure smooth playback with better RTP performance, try utilizing the following options as shown below.

→in lwipopts.h,

PBUF_POOL_SIZE 20 => 40 (just an example)
PBUF_POOL_BUFSIZE 500 => 250
DEFAULT_UDP_RECVMBOX_SIZE 6 => 24 (just an example)

→in opt.h,

MEMP_NUM_NETBUF 2 => 24 (just an example)
TCPIP_MBOX_SIZE 6 => 6 (just an example)

2.12MP3 Play Audio from SDCARD

This highlights the use of the mp3 example for playing the mp3 media from SDCARD.

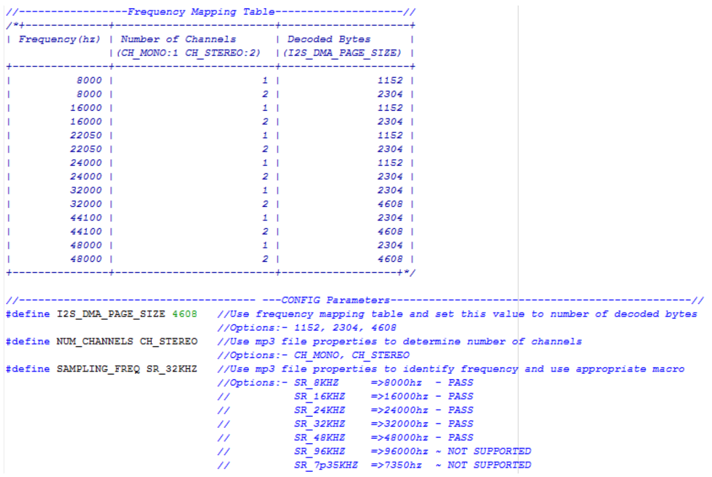
Pre-requisites

- Ameba board
- USB cable is used to connect with PC.
- ALC5651 Extension board or SGTL5000 Extension board.
- 3.5mm Audio jack to hear audio output.
- SDCARD sniffer in order to read SDCARD.

Procedures

- In platform_opts.h edit the macro "CONFIG_EXAMPLE_AUDIO_MP3" and set it to 1.
- Set the Config parameters in the beginning of the example file as shown below.

- The “I2S_DMA_PAGE_SIZE”, “NUM_CHANNELS” and “SAMPLING_FREQ” also need to be set appropriately.



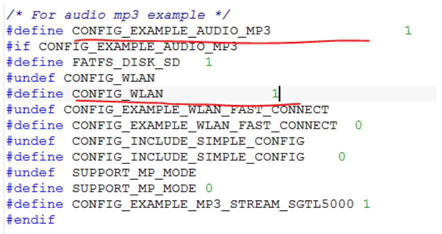
```
//----------------Frequency Mapping Table-------------------//
/*+-------------+-----------------------+-----------------+
| Frequency(hz) | Number of Channels    | Decoded Bytes   |
                |(CH_MONO:1 CH_STEREO:2) |(I2S_DMA_PAGE_SIZE) |
+-------------+-----------------------+-----------------+
|         8000 |                     1 |            1152 |
|         8000 |                     2 |            2304 |
|        16000 |                     1 |            1152 |
|        16000 |                     2 |            2304 |
|        22050 |                     1 |            1152 |
|        22050 |                     2 |            2304 |
|        24000 |                     1 |            1152 |
|        24000 |                     2 |            2304 |
|        32000 |                     1 |            2304 |
|        32000 |                     2 |            4608 |
|        44100 |                     1 |            2304 |
|        44100 |                     2 |            4608 |
|        48000 |                     1 |            2304 |
|        48000 |                     2 |            4608 |
+-------------+-----------------------+----------------+*/

//---------------------------------- ---CONFIG Parameters------------------------------------------//
#define I2S_DMA_PAGE_SIZE 4608   //Use frequency mapping table and set this value to number of decoded bytes
                                 //Options:- 1152, 2304, 4608
#define NUM_CHANNELS CH_STEREO   //Use mp3 file properties to determine number of channels
                                 //Options:- CH_MONO, CH_STEREO
#define SAMPLING_FREQ SR_32KHZ   //Use mp3 file properties to identify frequency and use appropriate macro
                                 //Options:- SR_8KHZ     =>8000hz  - PASS
                                 //          SR_16KHZ    =>16000hz - PASS
                                 //          SR_24KHZ    =>24000hz - PASS
                                 //          SR_32KHZ    =>32000hz - PASS
                                 //          SR_48KHZ    =>48000hz - PASS
                                 //          SR_96KHZ    =>96000hz ~ NOT SUPPORTED
                                 //          SR_7p35KHZ  =>7350hz  ~ NOT SUPPORTED
```

- As shown above the I2S_DMA_PAGE_SIZE should be set to the value of decoded bytes for the particular frequency and number of channels using the frequency mapping table that is mentioned in section 2.4.2 and also provided in the c file as comments.
- Rebuild the project and download and flash the binary onto the Ameba board.
- Once the software is downloaded, connect the SDCARD sniffer and the audio output board to the appropriate pins on the Ameba board.
- Load the mp3 file onto the SDCARD and rename the file as “AudioSDTest.mp3” this is very essential for the mp3 file to be detected properly by the software.
- Once the SDCARD is ready, insert it into the SD sniffer and press the reset button, the audio should start playing from the audio jack.

2.12.1 Details to build MP3 in GCC project

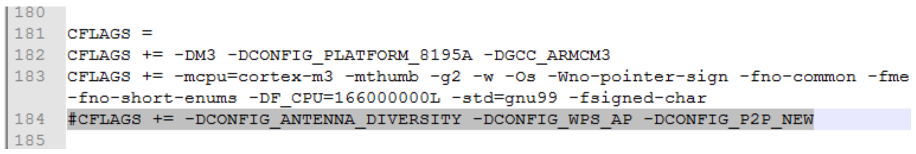
In order to be able to run the example for playing mp3 from SDCARD using the GCC project, the following points need to be noted.

- Enable the flag “CONFIG_EXAMPLE_AUDIO_MP3” in platform_opts.h and also enable WLAN if needed as shown below.



```
/* For audio mp3 example */
#define CONFIG_EXAMPLE_AUDIO_MP3               1
#if CONFIG_EXAMPLE_AUDIO_MP3
#define FATFS_DISK_SD   1
#undef CONFIG_WLAN
#define CONFIG_WLAN                  1
#undef CONFIG_EXAMPLE_WLAN_FAST_CONNECT
#define CONFIG_EXAMPLE_WLAN_FAST_CONNECT  0
#undef  CONFIG_INCLUDE_SIMPLE_CONFIG
#define CONFIG_INCLUDE_SIMPLE_CONFIG    0
#undef  SUPPORT_MP_MODE
#define SUPPORT_MP_MODE 0
#define CONFIG_EXAMPLE_MP3_STREAM_SGTL5000 1
#endif
```

- In case you are using the WLAN along with mp3 codec then the “CONFIG_ANTENNA_DIVERSITY” flag must be disabled in the make file of wlan library as shown below. Comment the line in the makefile.



```

CFLAGS =
CFLAGS += -DM3 -DCONFIG_PLATFORM_8195A -DGCC_ARMCM3
CFLAGS += -mcpu=cortex-m3 -mthumb -g2 -w -Os -Wno-pointer-sign -fno-common -fme
-fno-short-enums -DF_CPU=166000000L -std=gnu99 -fsigned-char
#CFLAGS += -DCONFIG_ANTENNA_DIVERSITY -DCONFIG_WPS_AP -DCONFIG_P2P_NEW

```

- In order to be able to fit the whole mp3 codec and wlan in the GCC project, the following changes need to be made to the linker file “rlx8195A-symbol-v02-img2.ld”.

```
.sdr_text :
 {
   __sdram_data_start__ = .;
   *(.sdram.text*)
   *(.p2p.text*)
   *(.wps.text*)
   *(.websocket.text*)
 } > SDRAM_RAM

 .sdr_rodata :
 {
   *(.sdram.rodata*)
   *(.p2p.rodata*)
   *(.wps.rodata*)
   *(.websocket.rodata*)
		*lib_sdcard.a: (.rodata*)
		*lib_codec.a: (.rodata*)
 } > SDRAM_RAM

 .sdr_data :
 {
   *(.sdram.data*)
   *(.p2p.data*)
   *(.wps.data*)
   *(.websocket.data*)
		*lib_sdcard.a: (.data*)
		*lib_codec.a: (.data*)
   . = ALIGN(256);
   *(.fpb.remap*)
   . = ALIGN(4);
   __sdram_data_end__ = .;
 } > SDRAM_RAM

 .sdr_bss :
 {
   __sdram_bss_start__ = .;
   *(.sdram.bss*)
   *(.p2p.bss*)
   *(.wps.bss*)
   *(.websocket.bss*)
		*lib_sdcard.a: (.bss*)
		*lib_codec.a: (.bss*)
   __sdram_bss_end__ = .;
 } > SDRAM_RAM
```

As highlighted in the snippet above, the following additions need to be made to the SDRAM blocks to add the .data, .rodata and .bss sections of the mp3 codec and lib_sdcard to SDRAM. This will give enough memory to fit the WLAN lib. Also in order for this change to work, the SDRAM blocks should be moved above all the BD_RAM blocks in the linker file to ensure that the changes are reflected, for more details take a look at the wiki:
https://wiki.realtek.com/display/CN3SD9/GCC%3A+How+to+locate+one+object+file+in+different+section

- Once this is done, give the make command and build the project.

2.12.2 Some common issues in GCC project.

The GCC project for building mp3 codec might run into some common issues, some of them are listed below.

- Unable to find the lib_codec.

  In case the codec lib is not found, try making the individual library using the command “make –f lib_codec.mk” after which you can make the application

- The heap is not enough after flashing example with mp3 and WLAN.

  If the heap is not enough after flashing MP3 and WLAN, just enable the SDRAM heap from the file heap_5.c