USB Video Capture example

This document illustrates how to set up environment to facilitate usb camera streaming. Details on playing mjpeg video using rtsp/rtp mechanism is also provided for reference.

Table of Contents

1 Introduction

It requires special guide to turn on this function in SDK. Please contact FAE or agent for help.

Only several camera modules are included in supported vendor list, please contact FAE or agent for help. Before activation, please contact agent to apply for USB camera module.

This document briefs configuration imperative for usb host driver to enable video streaming based on UVC/V4L2 framework. UVC/V4L2 driver is provided for video capturing and rtsp/rtp server for streaming out via wifi if needed. Following sections illustrate how several configurations are made to enable UVC streaming.

2 UVC Test Configuration

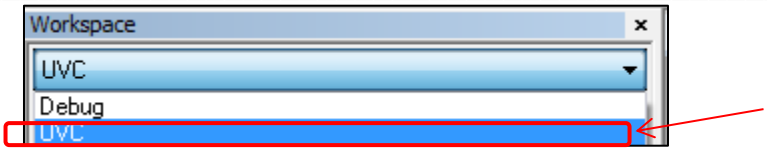
2.1 Configuration For Video Capture

UVC example is provided in component\common\example and is workable given the requisite that usb host driver is included and enabled in workspace. Since usb lib will be provided, just set usb relevant flags as below in platform_autoconf.h:

```
#define CONFIG_USB_EN 1
#undef  CONFIG_USB_NORMAL
#define CONFIG_USB_TEST 1
#define CONFIG_USB_MODULE 1
#define CONFIG_USB_VERIFY 1
//#define CONFIG_USB_DBGINFO_EN 1
#undef  DWC_DEVICE_ONLY
#define  DWC_HOST_ONLY 1
#define CONFIG_USB_HOST_ONLY 1
```

Several setting should be made in order to run uvc example test successfully.

1. When open project in IAR, go straight to workspace window and you will find a folded menu on top. Unfold the menu and click on UVC option to make life easier because we already make necessary configurations for UVC to run smoothly in this mode by defining CONFIG_UVC flag.



2. As uvc requires large-size buffers to store and deliver video data, it is necessary we assign enough heap size to cater to uvc example. You may modify total heap size by changing the value of configTOTAL_HEAP_SIZE in freeRTOSConfig.h. Recommended value should be at least 110kB.

```
#define configMINIMAL_STACK_SIZE                ( ( unsigned short ) 70 )
#ifdef CONFIG_UVC
#define configTOTAL_HEAP_SIZE                   ( ( size_t ) ( 110 * 1024 ) )
#else
#define configTOTAL_HEAP_SIZE                   ( ( size_t ) ( 60 * 1024 ) )
#endif
```

3. CONFIG_EXAMPLE_UVC is the flag to turn on/off uvc streaming example and can be modified in platform_opts.h. Make sure CONFIG_EXAMPLE_UVC flag is set to 1 so that uvc example code can be linked for use. If you have done step 1 you can ignore this check.

```
/* For uvc example */
#ifdef CONFIG_UVC
#define CONFIG_EXAMPLE_UVC                  1
#else
#define CONFIG_EXAMPLE_UVC                  0
#endif
```

2.2 Basic UVC api Specifics

UVC user api is declared in uvc_intf.h.

i. void uvc_stream_init(void)

description:
entry function to start uvc driver. Must be called at the very beginning before streaming.
argument : null.
return value: void.

ii. void uvc_stream_free(struct stream_context *stream_ctx)

description:
function to release all uvc streaming related resources. Must be called after camera streaming off (i.e. after calling uvc_stream_off()).
argument: pointer to struct stream_context type (refer to rtsp_api.h for details)

return value: void.

iii. int uvc_stream_on(struct stream_context *stream_ctx)

description:
function to turn on video capture and enable usb camera processing.
argument: pointer to struct stream_context type
return value: int type. Return 0 if success and negative values if fail.

iv. void uvc_stream_off(struct stream_context *stream_ctx)

description:
function to turn off video capture and disable usb camera processing.
argument: pointer to struct stream_context type
return value: void.

v. int uvc_set_param(struct stream_context *stream_ctx, uvc_fmt_t fmt_type, int width, int height, int frame_rate)

description:
function to set streaming video preference, including video format type, resolution and frame rate.
argument: pointer to struct stream_context type
uvc_fmt_t type: currently support UVC_FORMAT_MJPEG only
int type: width of capturing image
int type: height of capturing image
int type: frame rate of streaming video
return value: int type. Return 0 if success and negative values if fail.

vi. int uvc_buf_check(struct uvc_buf_context *b)

description:
function to check legality of uvc_buf_context. It is safe to do the check after using uvc_buf_context to retrieve internal uvc buffer information by calling uvc_dqbuf().
argument: struct uvc_buf_context type (refer to uvc_intf.h for details)
return value: int type. Return 0 if legal otherwise return negative value.

vii. int uvc_dqbuf(struct stream_context *stream_ctx, struct uvc_buf_context *b)

description:
function to dequeue uvc buffer and retrieve corresponding information for data processing.
argument: pointer to struct stream_context type
pointer to struct uvc_buf_context type
return value: int type. Return 0 if success otherwise return negative value.
NOTE: b->index will be set to -1 if empty uvc buffer is retrieved

viii. int uvc_qbuf(struct stream_context *stream_ctx, struct uvc_buf_context *b)

description:

function to queue corresponding uvc buffer back for new data.
argument: pointer to struct stream_context type
pointer to struct uvc_buf_context type
return value: int type. Return 0 if success otherwise return negative value.

ix. int is_pure_thru_on(struct stream_context *stream_ctx)
description:
function to check if driver enters pure throughput test mode. RTSP streaming will fail in this mode for video decoding is disabled.
argument: pointer to struct stream_context type
return value: in type. Return 1 if pure throughput test mode is on otherwise return 0.

x. ***void uvc_pure_thru_on(struct stream_context *stream_ctx)***
description:
function to turn on uvc pure throughput log service.
argument: pointer to struct stream_context type
return value: void.

xi. void uvc_dec_thru_on(struct stream_context *stream_ctx)
description:
function to turn on uvc throughput log service with video payload decoding.
argument: pointer to struct stream_context
return value: void.

xii. ***void uvc_thru_off(struct stream_context *stream_ctx)***
description:
function to turn off uvc throughput log service.
argument: pointer to struct stream_context
return value: void.

3.3 UVC throughput test configuration

UVC driver provide clients with two mutual exclusive modes for UVC throughput log service. Call uvc_pure_thru_on() to enter pure throughput test mode. UVC driver will keep record of raw data received by urb and dump it in a periodical manner. Be noted that since uvc driver will disable video decoding in this mode, you cannot retrieve any data for live streaming or image processing. However you can opt uvc_dec_thru_on() to allow video decoding in UVC layer while keeping an eye on throughput statistics. Call uvc_thru_off() if you don't want to show throughput information in console any more. UVC throughput test is turned off by default.

You will see UVC throughput log info as below:

```
start pure thru log
uvc thru:1348kB/s
uvc thru:1064kB/s
uvc thru:1057kB/s
uvc thru:887kB/s
uvc thru:964kB/s
uvc thru:1079kB/s
uvc thru:1349kB/s
start thru log with decoding
uvc thru:942kB/s
uvc thru:1206kB/s
uvc thru:1063kB/s
uvc thru:1058kB/s
uvc thru:1347kB/s
uvc thru:1047kB/s
uvc thru:1073kB/s
uvc thru:1346kB/s
stop thru log
```

You should be able to run basic uvc example at your preference after above configurations are completed.

3 UVC wifi streaming Configuration

UVC example allows combination with wlan driver to implement video streaming out for live video playing in application layer. Video streaming can be sent either in raw data or wrapped up by network transport protocol (e.g. HTTP, RTSP). In this example we utilize RTSP/RTP mechanism to deal with wifi streaming. To turn on wifi streaming implementation, please check if the following setting has been made.

1. set UVC_RTSP_EN flag to 1 (in example_uvc.h) as displayed below (it is set to 1 by default in UVC work space):

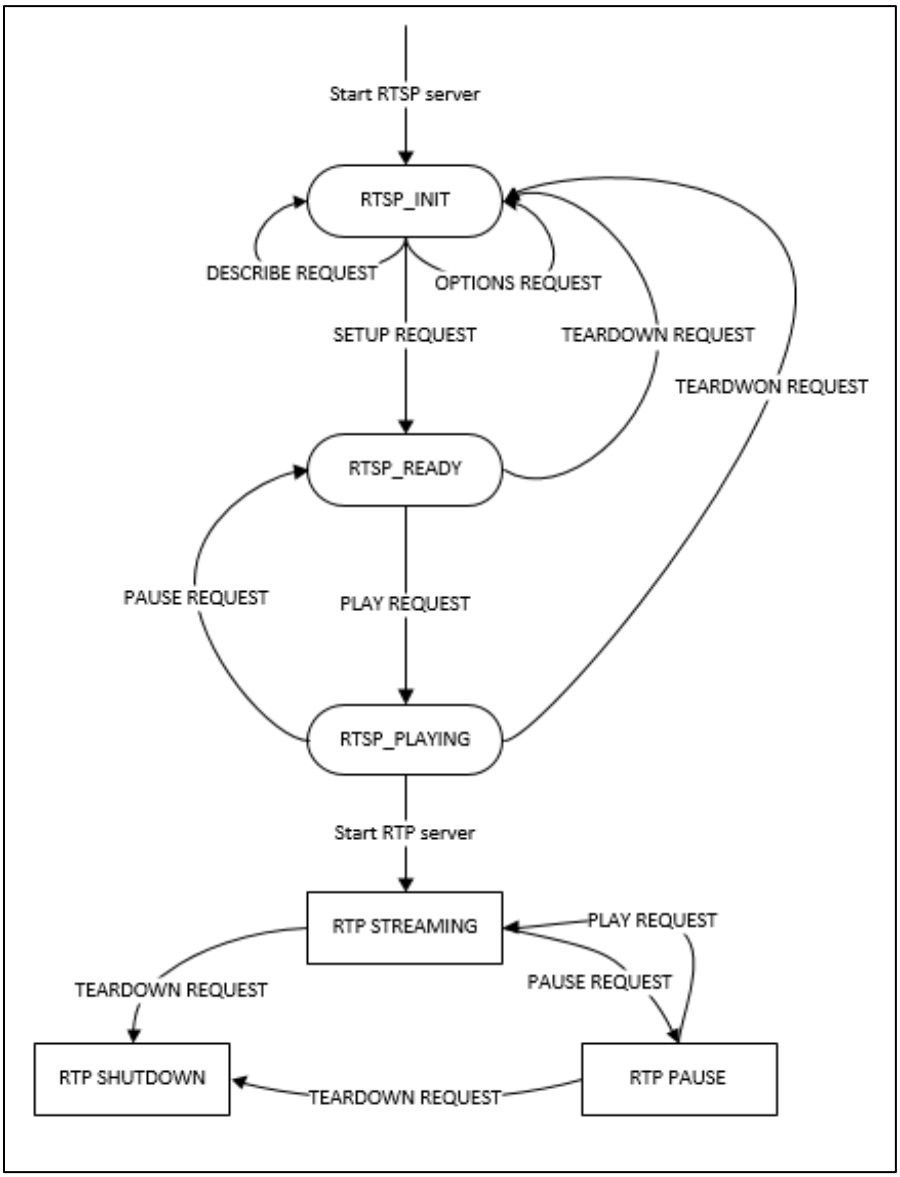
```
#define UVC_RTSP_EN      1
```

2. Ensure lwip and wlan are enabled for connecting to AP and implementing tcp/udp.
3. Make sure ameba connects to AP before RTSP server initialization. This is because RTSP server needs host ip address for creating socket before setting up connections with client. Fast connect method is recommended to get IP address from the beginning. In other cases, you might need to set enough delay time in uvc_rtsp_init() and manually connect to router using AT command.

Once UVC_RTSP_EN has been set, uvc_rtsp_init() will be called in uvc example which will create RTSP server to listen if any client requests to receive video stream. Accepted client will set up TCP connection with ameba RTSP server for streaming-use information exchange. After RTSP server receives a series of correct requests and sends corresponding responses, it will start an RTP server to stream video data out. RTSP server state machine is displayed in img_1.

The quality of UVC streaming is dependent of video format, resolution, framerate and even wlan TX performance to some extent. For video_oriented factors several APIs have been provided above for flexibility and for wlan TX you can adjust MAX_SKB_BUF_NUM value as large as possible within resource capacity to enhance its performance pertaining streaming live video. MAX_SKB_BUF_NUM can be accessed in wifi_skbuf.c in /network/api/wifi directory.

```
#undef MAX_SKB_BUF_NUM
#define MAX_SKB_BUF_NUM 18
```



Img_1. RTSP state machine

4 UVC wifi streaming on VLC

The procedure to start uvc wifi streaming on VLC is as follows:

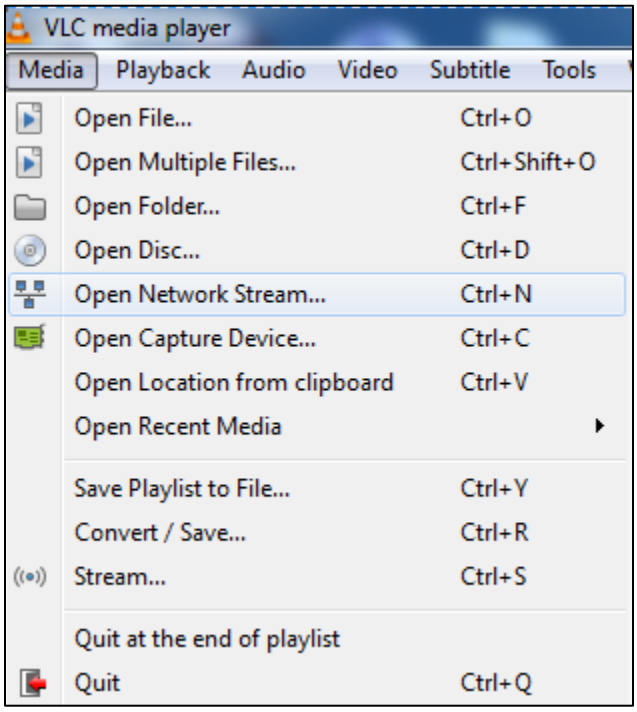
Step 1. Start ameba and wait until it connects to an AP nearby. Record its IP address. You may also use ATW? to get detailed wlan information.

```
# [ATW0]: _AT_WLAN_SET_SSID_ [bonjour]
[ATWC]: _AT_WLAN_JOIN_NET_

Joining BSS by SSID bonjour...

RTL8195A[Driver]: set ssid [bonjour]

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

RTL8195A[Driver]: auth success, start assoc

RTL8195A[Driver]: association success(res=3)

Connected after 98ms.

IP address : 192.168.0.121

Got IP after 611ms.


WIFI initialized
```

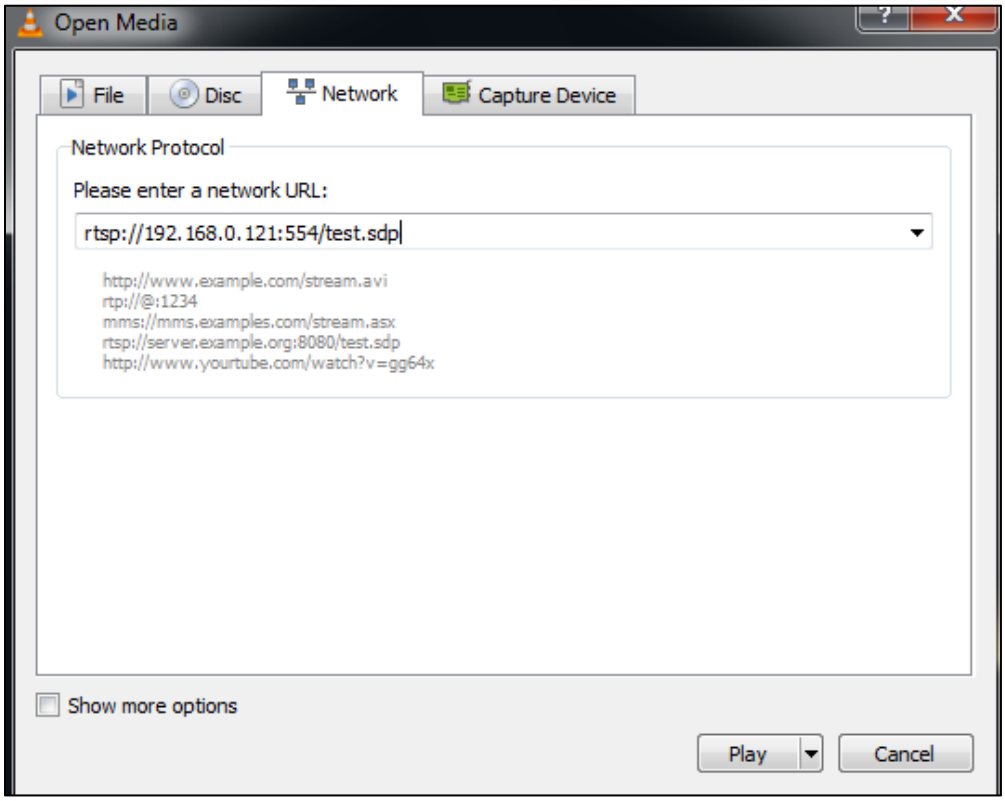
Step 2. Wait until rtsp server is ready. This is to guarantee ameba connect to AP before rtsp server tries to obtain correct IP address.

```
rtsp context initialized! addr: 0x1005d584
```

Step 3. Tune PC in the same network as ameba is. You may do ping test to check if they can communicate with each other successfully. Then open VLC and choose ‘open network stream’ in media menu.



Step 4. Switch to network window and type in: ‘rtsp://xxx.xxx.xxx.xxx:**//test.sdp’ before clicking play button. xxx.xxx.xxx.xxx refers to ameba IP address and ** is rtsp server port number (default is 554). You can tick ‘Show more options’ box for advanced settings.


Open Media
File
Disc
Network
Capture Device
Network Protocol
Please enter a network URL:
rtsp://192.168.0.121:554/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.yourtube.com/watch?v=gg64x
Show more options
Play
Cancel