TI Designs

SimpleLink™ CC3200-OV788 Video and Audio Streaming Over Wi-Fi® Reference Design

Description

This TI Design combines TI wireless technology with OmniVision™'s AV technology to bring live streaming capabilities of audio and video data over Wi-Fi®.

Resources

TIDC-CC3200-VIDEO	Design Folder
CC3200MOD	Product Folder
SN74AVC8T245PW	Product Folder
SN74AVC4T245PW	Product Folder
SN74AVC2T45DRL	Product Folder
SN74AVC1T45DRLR	Product Folder

TI E2E™ Community

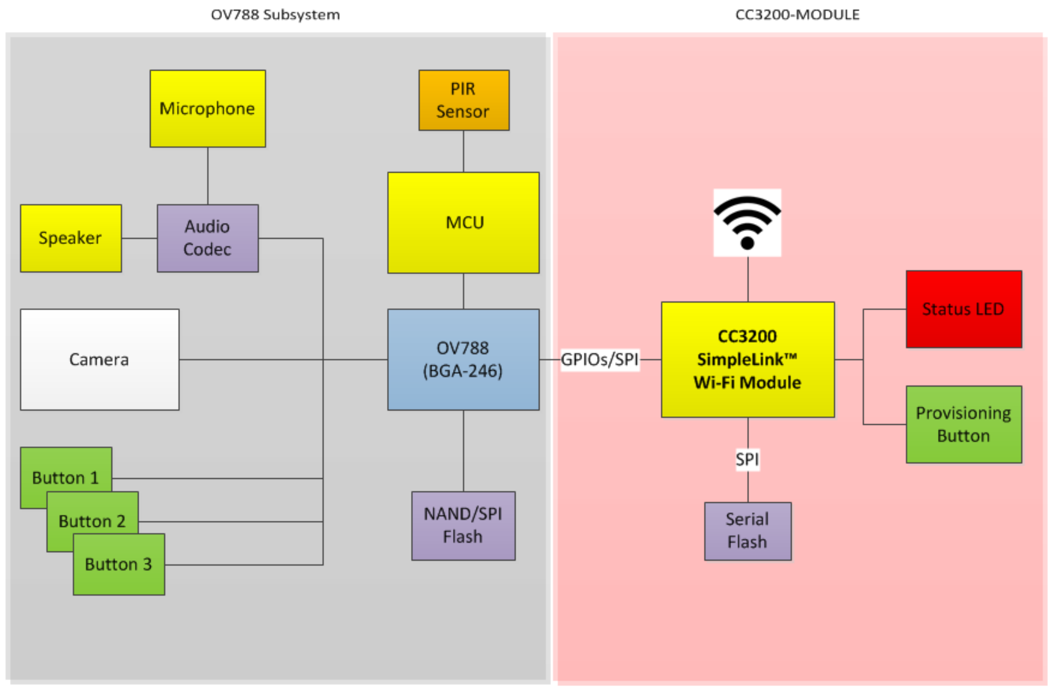
ASK Our E2E Experts

Features

- Supports capture and streaming of:
  - Video – 720p at 15FPS
  - Audio – PCM, 16 bps at 11025 Hz
- Supports RTP/RTSP protocols – Compatible with media players supporting these protocols
- SimpleLink™ Wi-Fi connectivity over 802.11 b/g/n networks from any smart phone, tablet, or computer over local network
- Built-in provisioning for easy commissioning of the device to a Wi-Fi network
- Advanced low-power modes

Applications

- Wireless Video Doorbells
- Home and Baby Monitoring
- Remote Surveillance
- Wireless Announcement Systems



1 System Overview

1.1 System Description

The TIDC-CC3200-VIDEO TI Designs reference design combines TI wireless technology with OmniVision AV technology to enable live streaming of audio and video data over Wi-Fi. The exponential increase in the demand for Wi-Fi-enabled battery-operated cameras, and for applications ranging from video doorbells to surveillance devices, has fueled the need for reference designs that enable developers to quickly add video and audio functionalities to their end applications.

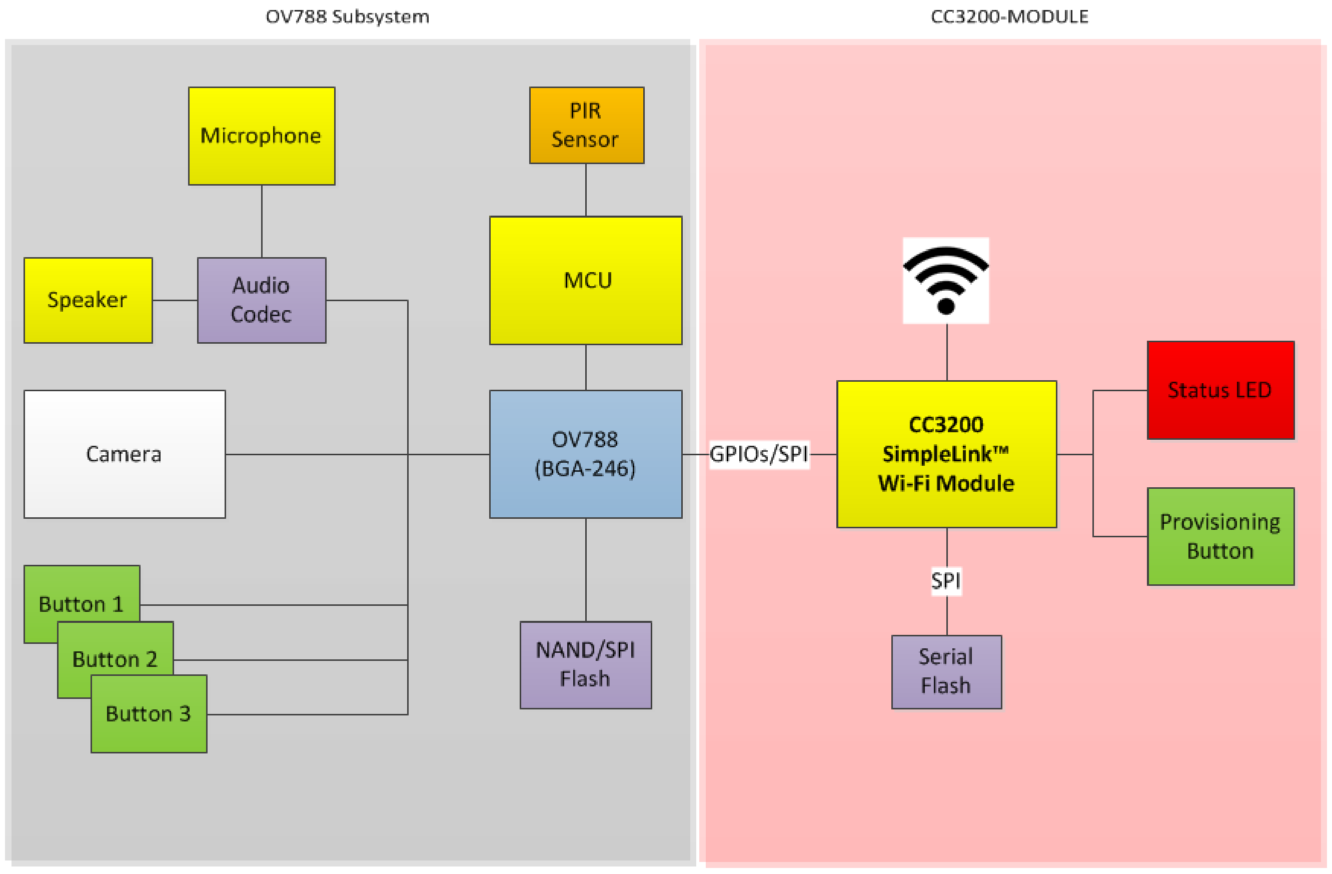
This design provides an integrated solution showcasing the CC3200's ability to provide a full system solution for audio-video streaming applications. On boot-up, the application initializes the OV788 subsystem, configures the CC3200 to connect to a Wi-Fi network, opens an RTSP server, and waits for the RTSP clients, such as media player applications, to connect and request for live streams.

1.2 Key System Specifications

Table 1. Key System Specifications

PARAMETER	SPECIFICATIONS	
Capture and Streaming Quality	Video	720p at 15FPS
	Audio	PCM, 16 bps at 11025 Hz
Featured Protocols	RTP/RTSP	
Supported Networks	802.11 b/g/n	

1.3 Block Diagram



1.4 CC3200MOD

Created for the Internet of Things (IoT), the SimpleLink™ CC3200MOD is a wireless MCU module that integrates an ARM Cortex®-M4 MCU, allowing customers to develop an entire application with a single device. With on-chip Wi-Fi, Internet, and robust security protocols, no prior Wi-Fi experience is required for fast development. The CC3200MOD integrates all required system-level hardware components, including clocks, SPI flash, RF switch, and passives, into an LGA package for easy assembly and low-cost PCB design.

1.4.1 Hardware Description

The CC3200 video camera hardware is divided into two subsystems:

- Wi-Fi and host subsystem
- Compressing and sensor subsystem – OVI

1.4.1.1 Wi-Fi and Host Subsystem

The CC3200 Wi-Fi board is a compact design that enables a video camera design with a small form-factor. The SN74AVC4T245PW level shifter is used to interface the 3.3-V CC3200 signals with the 1.8-V signals of the OV788.

1.4.1.1.1 Antenna Connection

The CC3200 Wi-Fi board design does not include an on-board antenna. An antenna can be attached using the UFL connector, as shown in Figure 1.

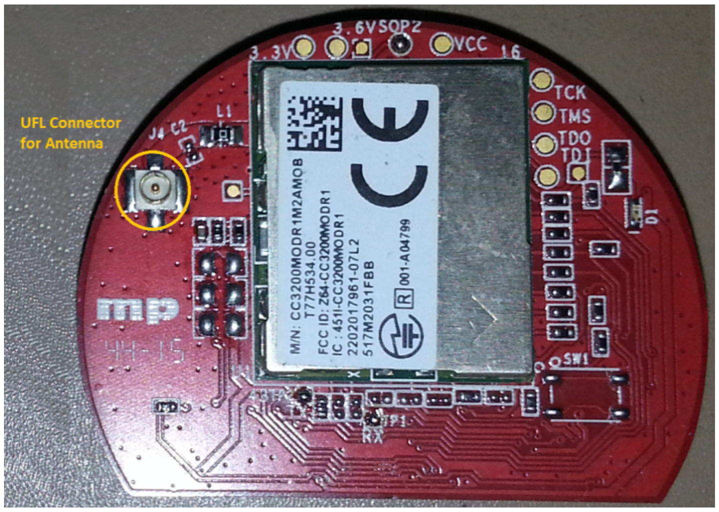


Figure 1. CC3200-MODULE UFL Connector Location

1.4.1.1.2 Programming Interface

Test points are available on the CC3200 Wi-Fi board for programming and debugging. These interfaces are showcased in Figure 2.

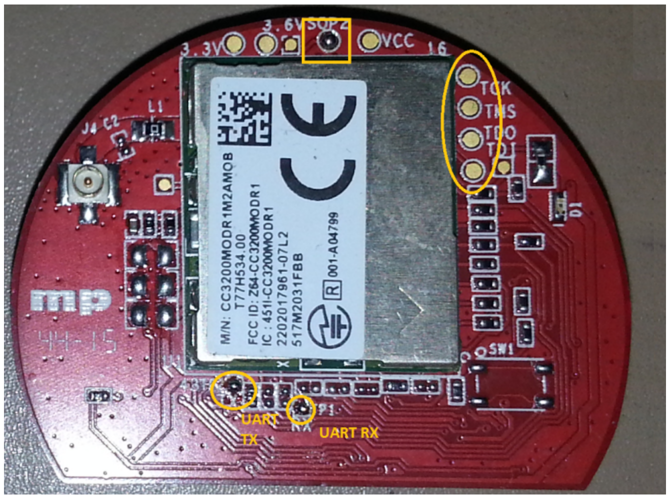


Figure 2. CC3200-MODULE Programming Interface

1.4.1.2 *Compressing and Sensor Subsystem - OVI*

OmniVision's OV9712 CMOS image sensor and OV788 video compression module are used for HD video streaming. To run the demo, configure the OV788 in boot 1101 mode. For detailed information regarding the OV788 hardware, contact OmniVision customer support (http://www.ovt.com/).

1.4.2 Connection Diagram

Table 2 and Figure 3 illustrate the connections between the CC3200 and the OV788.

Table 2. Description of Connections Between CC3200 and OV788 Subsystems

Connection	Description
SPI(CS, CLK, MISO, MOSI)	A 4-wire SPI is used for data exchange between the CC3200 and the OV788.
OVT_SYNC (GPIOA)	This signal is controlled by the CC3200 and used to notify the OV788 when the CC3200 is ready to send or receive data. Refer to the Figure 5 for details.
OVT_RDY (GPIOB)	This signal is controlled by the OV788 and used to notify the CC3200 when the OV788 is ready to receive or send data. Refer to the Figure 5 for details.
WLAN_ON	This signal is used to enable the 1A LDO. This signal should be asserted before enabling the NWP or powering on the OV788 subsystem.
POWER_EN	This signal is used to power on the OV788 subsystem.

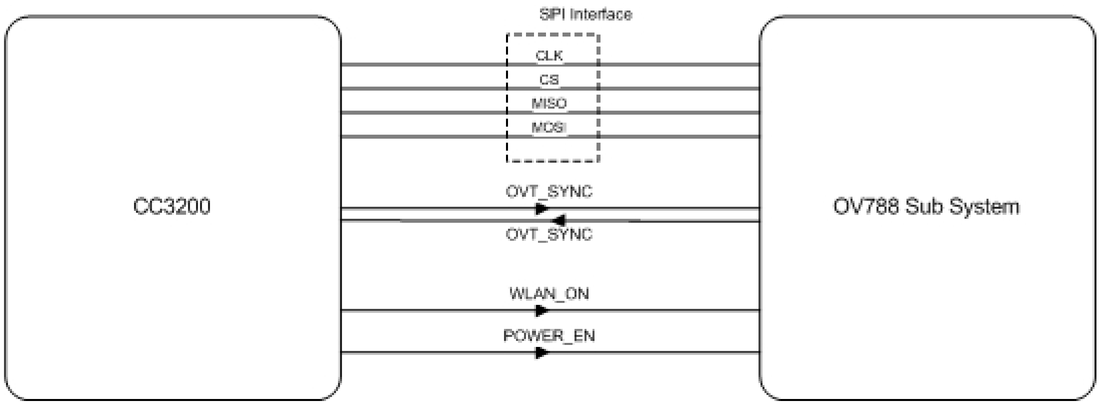


Figure 3. Diagram of Connections Between CC3200 and OV788 Subsystems

1.4.3 Embedded Software System

The software architecture for the design is described in the following subsections. The first subsection describes the setup of various software modules. Subsequently, each module is described in detail. The software can be separated into multiple parts: core application, RTSP library, RTCP/RTP library, and OV788 interface module.

Figure 4 shows a block diagram of the TIDC-CC3200-VIDEO software.

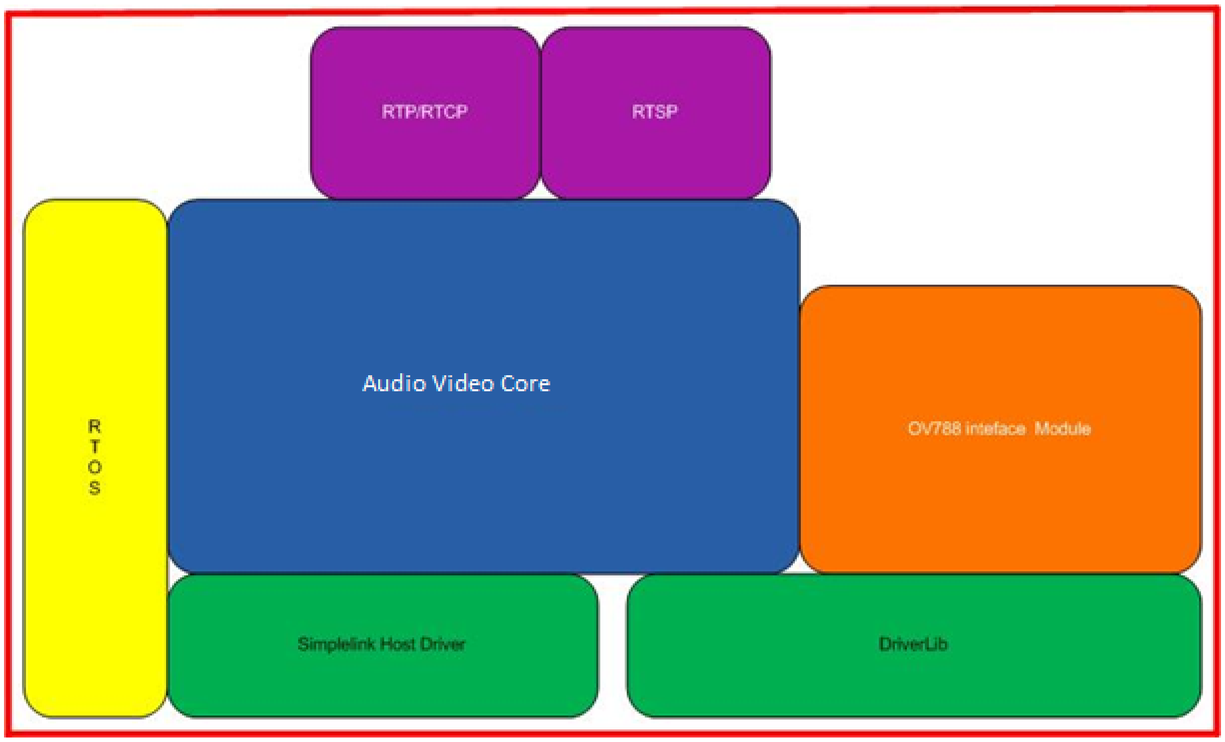


Figure 4. TIDC-CC3200-VIDEO Software Block Diagram

The core application is responsible for processing the incoming request, managing the connection, querying the audio and video data from OV788, and sending it over to the remote client using the RTSP/RTP/RTCP protocol.

1.4.3.1 OV788 Interface Library

The OV788 library contains the implementation for communicating with the OV788 module. Functions are provided to boot up the module, download the OV788 firmware, configure the sensor, and get the video and audio streams from the OV788.

The CC3200 uses the SPI interface (master mode) along with two GPIOs (SYNC and RDY) to communicate with the OV788, as shown in Figure 5.

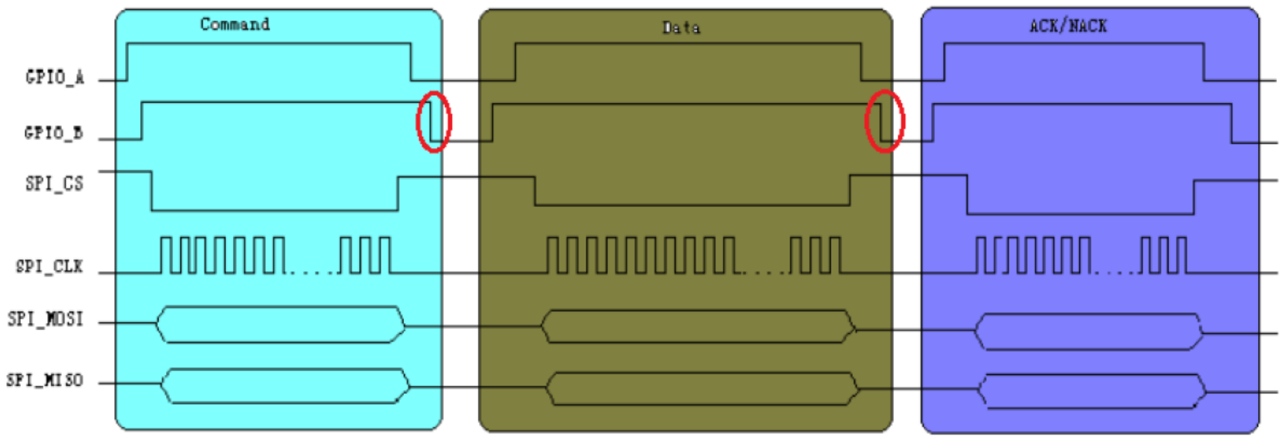


Figure 5. Interface Timing Diagram

The OV788 library provided with the package supports the following features:

- Boot-up and initialize OV788 module
- Power-off OV788
- Configure
- Frame rate
- Resolution
- Brightness
- Contrast
- Saturation
- Frequency
- Flip
- Mode
- Enable or disable video streaming
- Get video stream information
- Get video stream data
- Get audio stream information
- Get audio stream data

For the library API details, refer to the ov_sif_interface API document under <cc3200-sdk>/docs.

1.4.3.2 RTSP Library

The Real Time Streaming Protocol (RTSP) is used for establishing and controlling a media session between two systems. RTSP is generally used along with RTP/RTCP for media stream delivery.

The RTSP library provided contains an implementation of the RTSP server, used for processing the RTSP client request and generating the responses to be sent to the client. Table 3 lists the protocol request directives supported by the library.

Table 3. RTSP Library Requests

Request Directive	Description
OPTIONS	An OPTION request from the client to server is used to get the list of commands the server shall support or accept.
DESCRIBE	A DESCRIBE request contains the type of reply data that can be handled. The server reply contains the description of the media streams supported or controlled with the aggregated URL.
SETUP	A SETUP request specifies how a single media stream must be transported, along with the local ports for receiving RTP and RTCP data. This must be done before a PLAY request is sent by the client. The server reply confirms the chosen parameters, and adds the server's chosen ports.
PLAY	A PLAY request is used to start the streaming of media.
TEARDOWN	A TEARDOWN request is used to terminate the session. It stops all media streams and frees all session-related data on the server.

1.4.3.3 RTP/RTCP Library

The Real-Time Transport Protocol (RTP) is used for delivering the media over the network. RTP is used in conjunction with the RTP Control Protocol (RTCP). RTCP is used to monitor the transmission statistics and quality.

The provided RTP library supports the profile for H.264 Video, RFC6184, and is used to create the RTP and RTCP packets for the mentioned profile. It also supports PCM L16 audio.

1.4.3.4 Application Flow

Figure 6 shows the application flow.

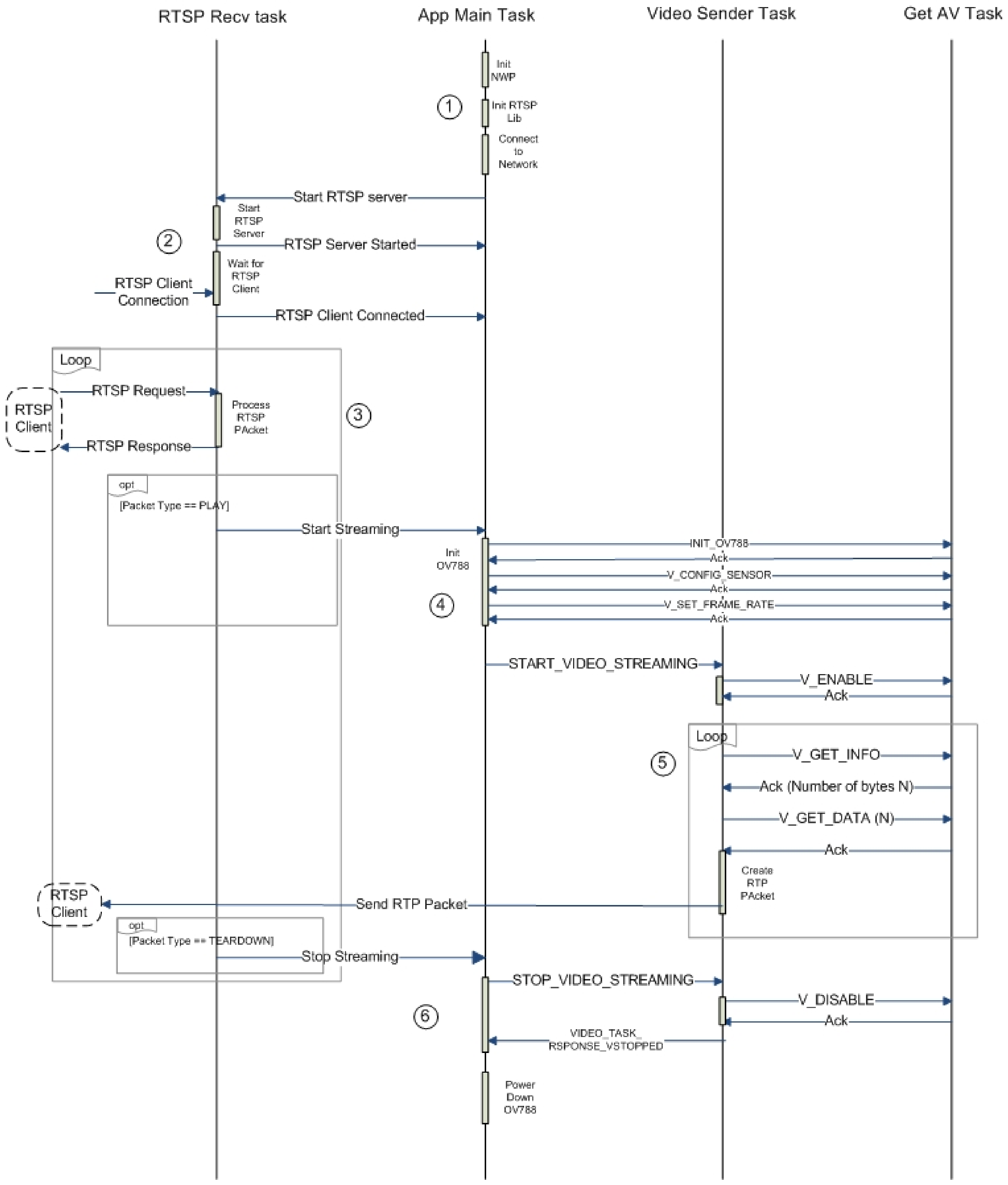


Figure 6. TIDC-CC3200-VIDEO Application Flow Diagram

1. The App Main Task initializes the network processor, initializes the RTSP library, and connects the device to a network.
2. When the device is connected to the network, the RTSP Recv Task starts a TCP server and waits for an RTSP client to connect.
3. When the client is connected and an RTSP packet is received, RTSP Recv Task processes the packet. If a PLAY packet is received, a message is sent to App Main Task to start the streaming. If a TEARDOWN packet is received, a message is sent to App Main Task to stop the streaming.
4. When a start streaming message is received, the App Main Task powers on the OV788 subsystem, downloads the OV788 firmware, and configures the sensor. Afterward, an initialization message is sent to the Video Sender Task and the Audio Sender Task to start the video and audio streaming, respectively.
5. When the message to start video streaming is received, the Video Sender Task sends the V_ENABLE message to enable the video on the OV788. After enabling, the Video Sender Task requests the available data info and, upon receiving valid data information, gets the video data in a loop until a stop streaming message is received. The same sequence is also followed for audio streams.
6. If the stop streamlining message is received by the App Main Task, a message is sent to the Video Sender Task and Audio Sender Task to stop streaming and wait for an acknowledgement message. Once the acknowledgement is received, the OV788 subsystem is powered down.

1.4.3.5 Low-Power Mode

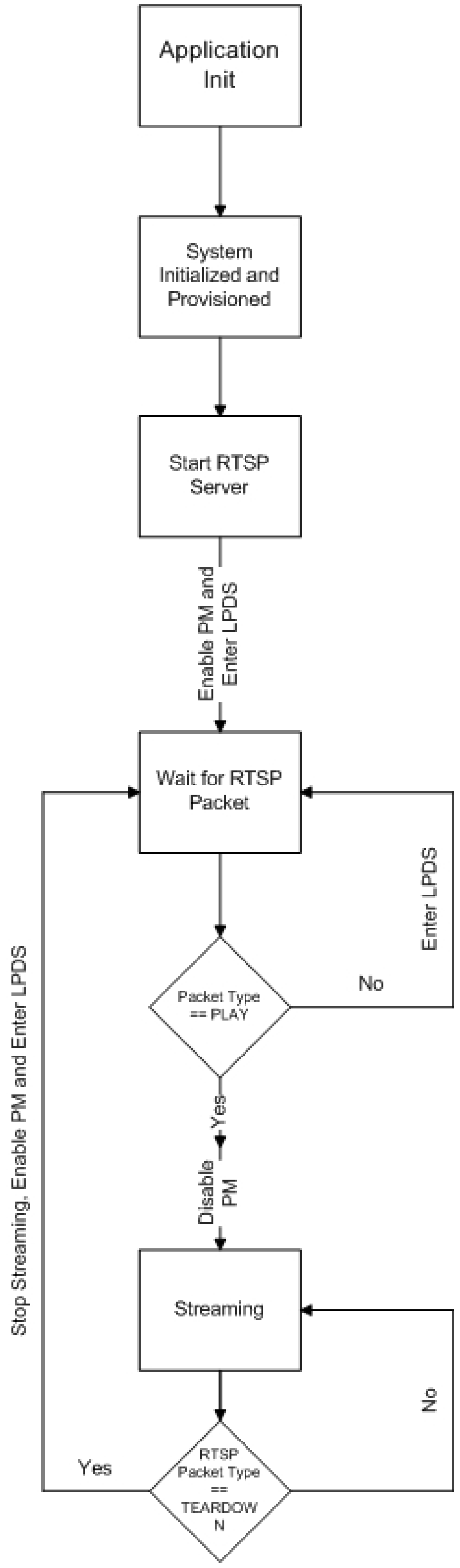


Figure 7. Application State Diagram

For low current consumption, while the system is not streaming the CC3200 MCU is configured in LPDS mode and the NWP is in idle connected mode. This power management scheme is selected to allow the remote app to connect at any time, and make a request for the video stream. Depending upon the use case, other low-power modes could be used (for example, if an intermediate server is used for storing and streaming the data with the CC3200 configured in client mode). The CC3200 client can be put in hibernate mode when idle, and can be woken up using an interrupt to start the streaming.

Using three AA Duracell batteries, the system can stream video continuously for approximately four hours. For static CC3200 current measurements across power modes, refer to the CC3200 data sheet (SWAS032).

Support for low-power mode is not enabled in this version of the application.

2 Getting Started

2.1 Prerequisites

The following components are required to evaluate the TIDC-CC3200-VIDEO design:

- CC3200 Module Rev 2.0
- OV788 Reference Design Board Rev 3.0
- OV9712 CMOS Camera Sensor with Optic Lens
- Apple® iPhone®/iPad® or Android™ Smartphone
- 802.11 b/g/n Wi-Fi Access Point
- Tools for Programming and Logging
  - CC3200-LAUNCHXL Rev 4.1
  - PC running Microsoft Windows® 7

2.2 Installation of Tools and Software

1. Download and install the following software components from ti.com to the Windows PC. This guide assumes the components are downloaded to location C:\TI.
   - CC3200SDK v1.2.0
   - CC3200SDK-PROVISIONING v1.0.0.0 (To be installed into the above SDK)
   - CC3200SDK-SERVICEPACK v1.0.1.6-2.6.0.5

   For installation and setup details of the SDK, refer to the CC3200-SDK's Getting Started Guide.
2. Download the CC3200 Video Doorbell package cc3200_video_doorbell_v01 from TI's shared location to the Windows PC and copy cc3200_video_doorbell_v01\cc3200-sdk\* to *C:\TI\CC3200SDK_1.2.0\cc3200-sdk\*. This action copies the following folders:
   - cc3200-sdk\docs to *C:\TI\CC3200SDK_1.2.0\cc3200-sdk\docs*
   - cc3200-sdk\example\video_camera to *C:\TI\CC3200SDK_1.2.0\cc3200-sdk\example*
   - cc3200-sdk\netapps\rtp_rtcp and cc3200-sdk\netapps\rtsp to *C:\TI\CC3200SDK_1.2.0\cc3200-sdk\netapps*
   - cc3200-sdk\third_party\ov_sif_inerface and cc3200-sdk\third_party\ov788_firmware to *C:\TI\CC3200SDK_1.2.0\cc3200-sdk\ third_party*
3. Install the provisioning application on the iOS/Android Smartphone.
4. Download the Uniflash tool from ti.com, and install on the Windows PC.
5. Download VLC v2.7 and Discovery iOS applications from Apple's App-Store, and install them on the iOS device. Alternatively, download the RTSP Player v4.4.0 and Bonjour Browser Android applications from Android's Play Store, and install them on the Android Smartphone

2.3 Updating Device Software

If the service pack, application binaries, and OV firmware files are already programmed on the CC3200MOD device, proceed to Section 2.4. Otherwise, follow the instructions below.

How to program the CC3200-MOD device:

1. Mount the CC3200-MODULE board on OV788 board as shown in Figure 8. Ensure J1 of both boards are aligned with each other.

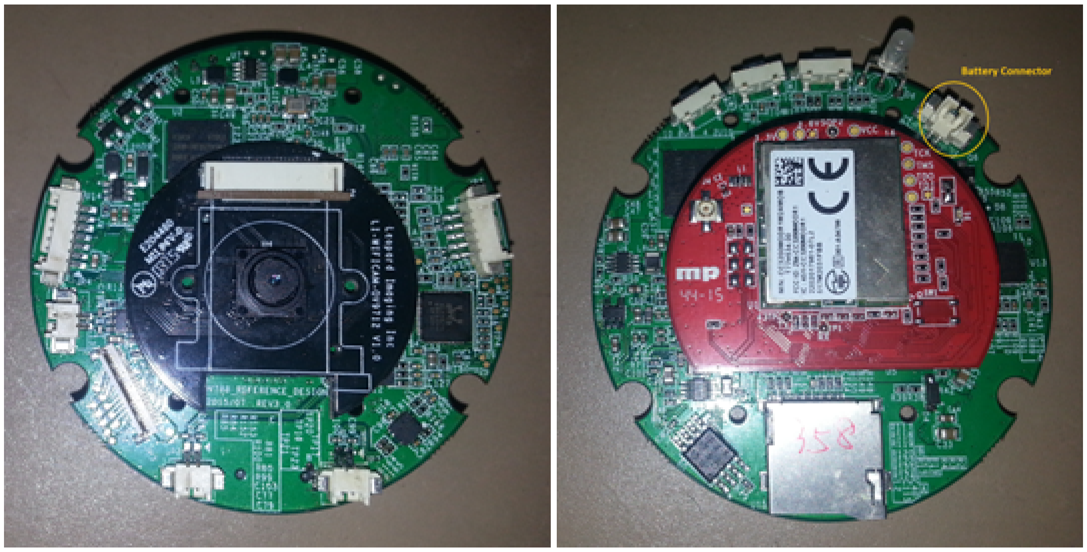


Figure 8. CC3200-MODULE Board Mounted on OV788 Subsystem

2. Assuming the jumpers on the CC3200-LAUNCHXL are connected as shown in Figure 9, remove jumpers J6, J7, J8, J9, J10, J11, J12, and J15.

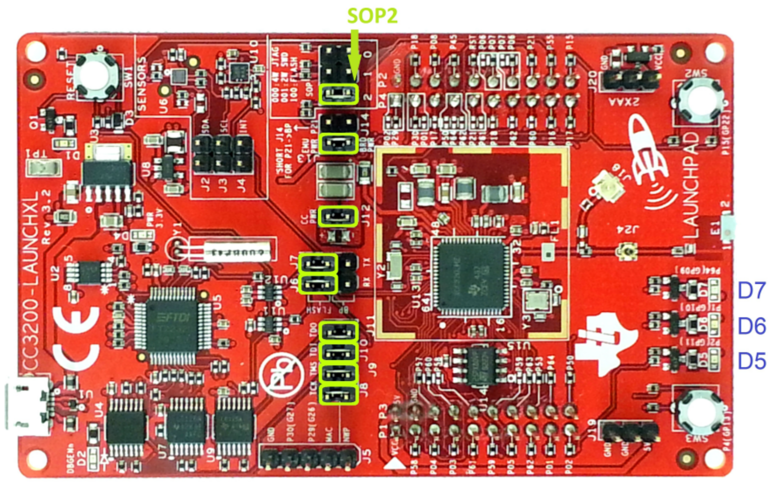
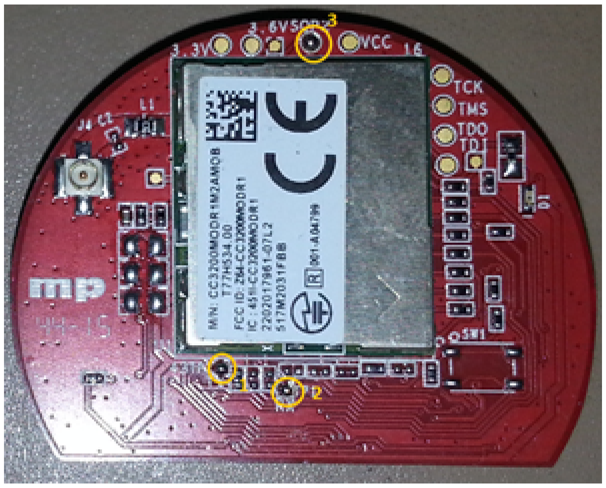


Figure 9. CC3200-LAUNCHXL Default Jumper Connections

3. Connect the UART TX line of the CC3200-MODULE board with Pin-1 of J7 on the CC3200-LAUNCHXL, as shown in Figure 10 [1].
4. Connect the UART RX line of the CC3200-MODULE board with Pin-1 of J6 on CC3200-LAUNCHXL, as shown in Figure 10 [2].
5. Connect the SOP2 line of the CC3200-MODULE board with VCC or Pin-1 of J15 on CC3200-LAUNCHXL, as shown in Figure 10 [3].



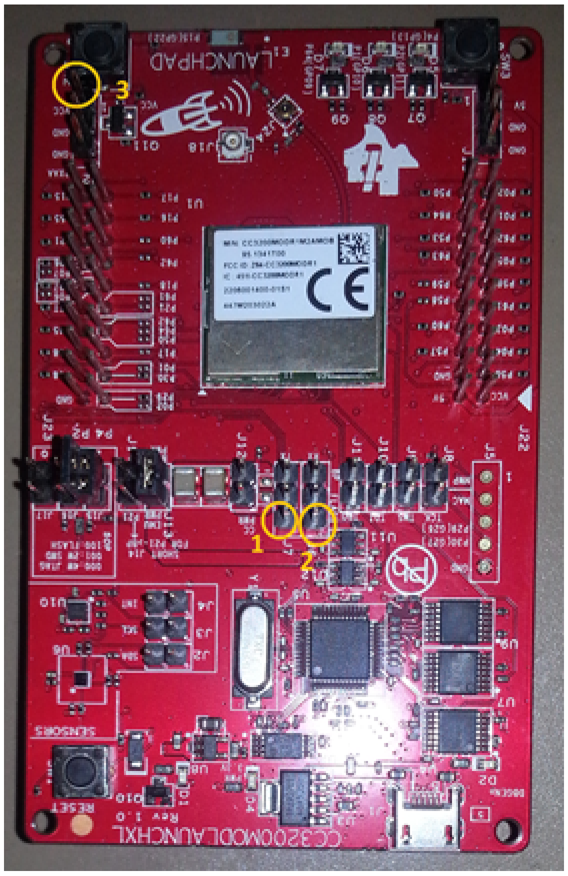


Figure 10. CC3200-MODULE to CC3200-LAUNCHXL Connection Locations

6. Connect J1 port of the CC3200-LAUNCHXL to the PC using a micro-USB cable. The device should be visible in the PC's Device Manager as shown in Figure 11, and a COM port shall be enumerated.

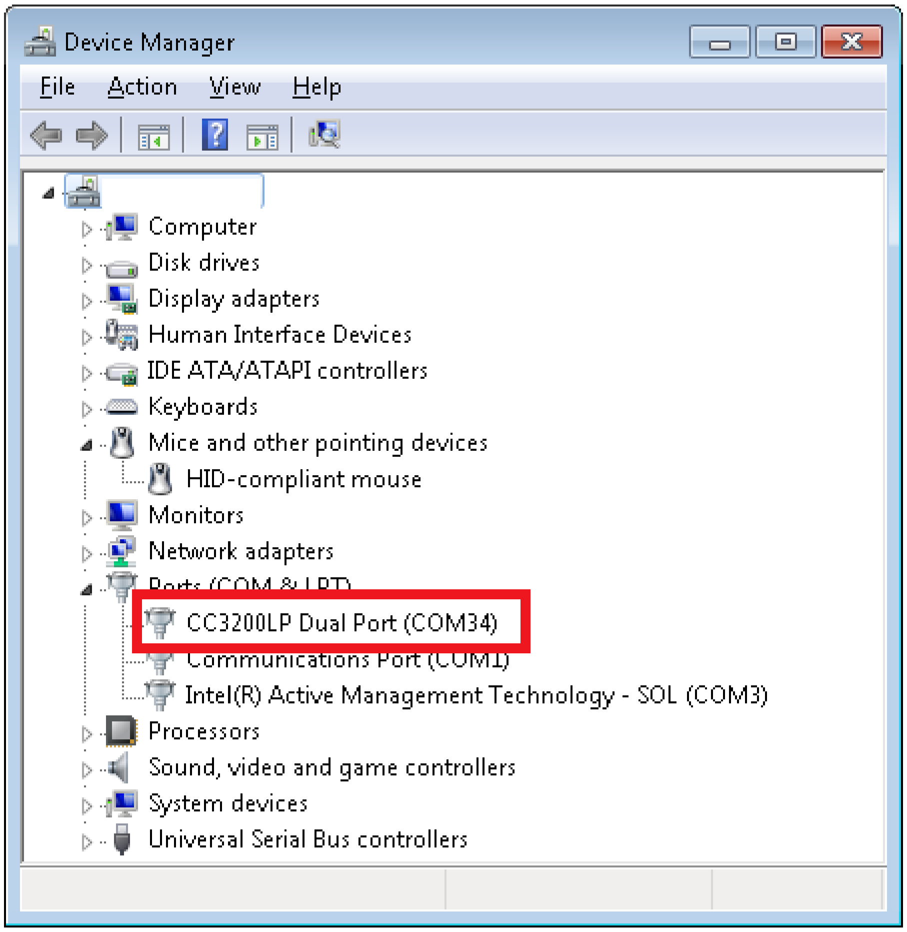

Device Manager
File Action View Help
Computer
Disk drives
Display adapters
Human Interface Devices
IDE ATA/ATAPI controllers
Keyboards
Mice and other pointing devices
HID-compliant mouse
Monitors
Network adapters
Ports (COM & LPT)
CC3200LP Dual Port (COM34)
Communications Port (COM1)
Intel(R) Active Management Technology - SOL (COM3)
Processors
Sound, video and game controllers
System devices
Universal Serial Bus controllers

Figure 11. CC3200 COM Port in Device Manager

7. Power up the OV788/CC3200-MODULE setup using the battery connector, as shown in Figure 8.
8. Launch the UniFlash tool and perform the following operations:
   (a) Use the Format button to format the serial flash on the CC3200-MODULE board.
   (b) Use the Service Pack Programming button to program Servicepack v1.0.1.6-2.6.0.5 on the CC3200-MODULE board.
   (c) Add a file to the configuration from the Operation -> Add File menu and name it "user/ovt_firmware.bin." For the URL, choose "dsif_slave.bin" from "cc3200-sdk/third_party/ov788_firmware." Check Erase, Update, and Verify.
   (d) Use the Program button to program the OV firmware and application binary on the CC3200-MODULE board.

NOTE: The CC3200-MODULE board does not have a RESET button. Pull out the power and plug it back in when a reset is required during the programming process.

NOTE: If a new Gen-1 provisioning is to be used, build simplelink_extlib/provisioninglib/ewarm/Release/Exe/provisioninglib.a with SL_PLATFORM_MULTI_THREADED defined, and the path set to $PROJ_DIR$/../../../oslib/ included. Also, rebuild simplelink.a.

2.4 Setting-up the Demonstration

2.4.1 Board Configuration

1. Mount the CC3200-MODULE board on the OV788 board as shown in Figure 12. Ensure J1 of both boards are aligned with each other.

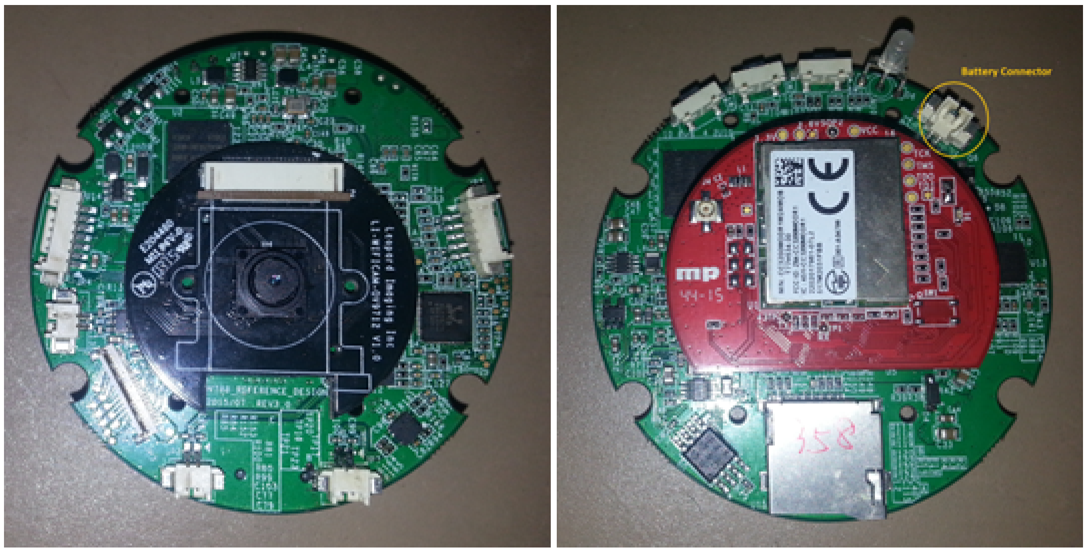


Figure 12. Assembled CC3200-MODULE and OV788 System

2. Power up the OV788/CC3200-MODULE setup using the battery connector shown in Figure 12. On powering up:
   - The application waits for about 10 seconds to establish a connection with a known AP. This state is indicated by a blinking red LED on the CC3200-MODULE board.
   - If the device fails to connect with a known AP or if there are no saved profiles, the device enters provisioning mode. This state is indicated by a solid red LED. Use the SimpleLink Starter Pro or Wi-Fi Starter Smartphone application to configure the device.
   - The red LED turns off if the connection is successfully established.
   - The application then initializes the OV788 board, configures the audio and video modules, starts the RTSP server, and listens for client connections.

2.4.2 AV Streaming With iPad and iPhone

1. Ensure that the applications listed in Section 2.2 are installed on an iPad or iPhone.
2. Connect the Smartphone or tablet to the same network that the demo device is connected to.
3. Connect a headphone using the 3.5-mm jack.
4. Open the Discovery application and navigate to local -> World Wide Web HTTP -> xxxxxxxxxxxx@mysimplelink, as shown in Figure 13.

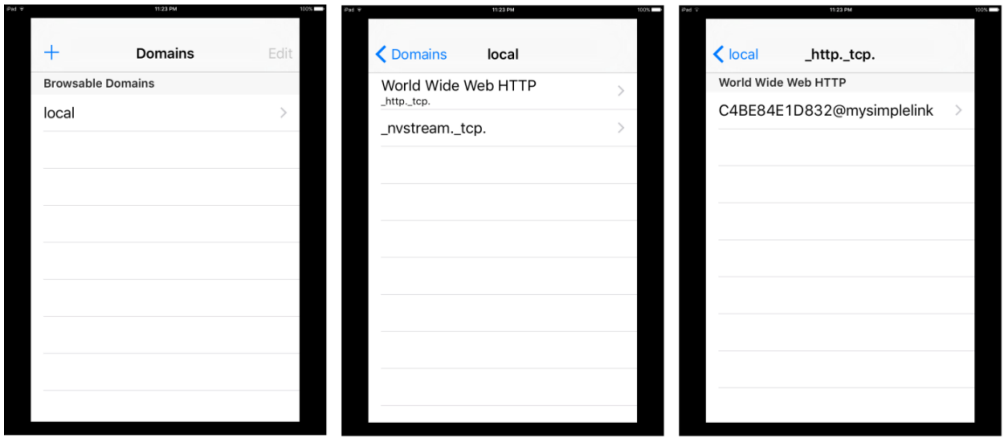
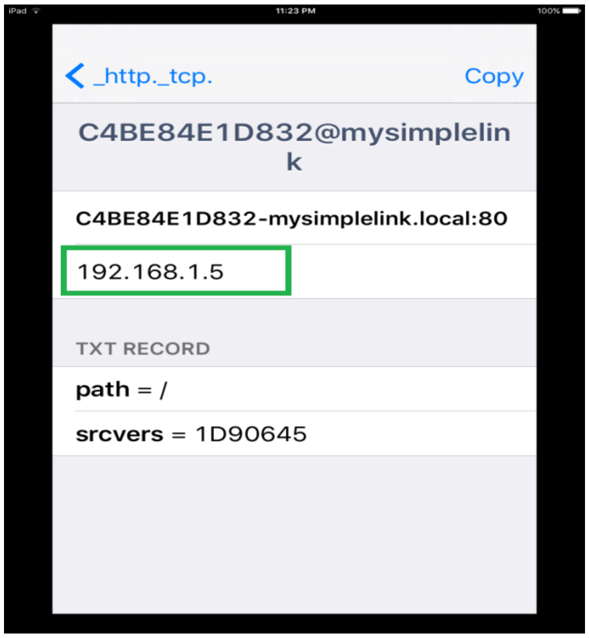


Figure 13. Discovery Application Navigation to XXXXXXXXXXXX@mysimplelink

5. Note the IP address of the CC3200, as shown in Figure 14.



C4BE84E1D832@mysimplelink

C4BE84E1D832-mysimplelink.local:80

192.168.1.5

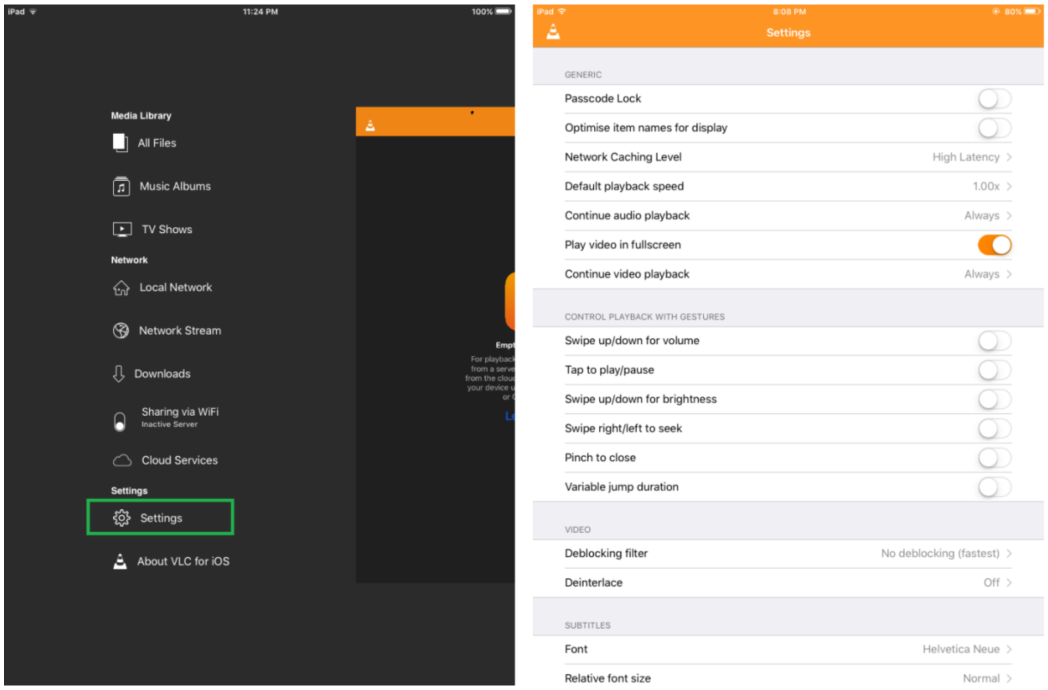
TXT RECORD

path = /

srcvers = 1D90645

Figure 14. IP Address Location in Discovery Application

6. Open the VLC application and navigate to the Settings tab, and use the settings shown in Figure 15.



Settings

GENERIC	
Passcode Lock	○
Optimise item names for display	○
Network Caching Level	High Latency
Default playback speed	1.00x
Continue audio playback	Always
Play video in fullscreen	●
Continue video playback	Always
CONTROL PLAYBACK WITH GESTURES	
Swipe up/down for volume	○
Tap to play/pause	○
Swipe up/down for brightness	○
Swipe right/left to seek	○
Pinch to close	○
Variable jump duration	○
VIDEO	
Deblocking filter	No deblocking (fastest)
Deinterlace	Off
SUBTITLES	
Font	Helvetica Neue
Relative font size	Normal

Figure 15. VLC Application Settings Tab

7. Navigate to the Network Stream tab, as shown in Figure 16.

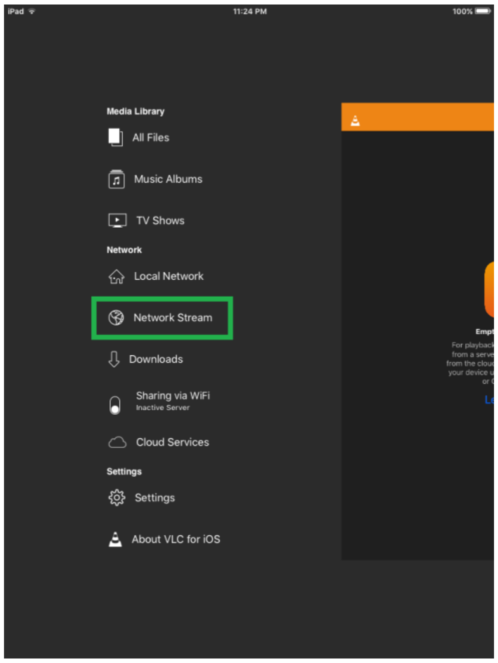


Figure 16. VLC Application Network Stream Tab

8. Enter rtsp://<demo-device-ip>:8554, and press Open Network Stream, as shown in Figure 17.

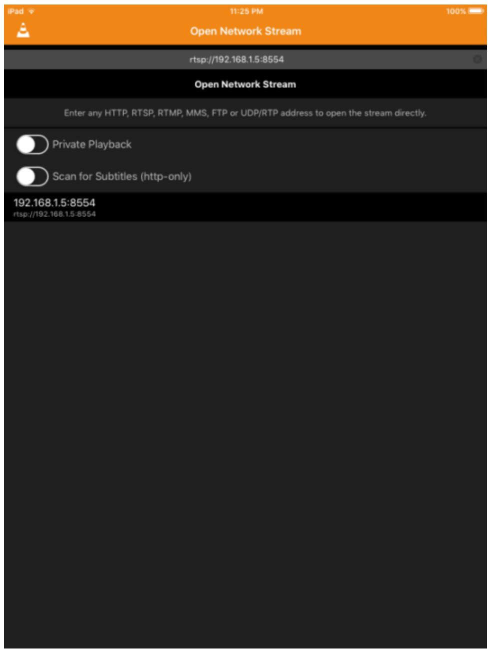


Figure 17. VLC Application Open Network Stream

9. The VLC application connects to the device, and starts playing the live video and audio streams.

2.4.3 V Streaming with Android Smartphone

1. Ensure that the applications listed in Section 2.2 are installed on the Android Smartphone.
2. Connect the Smartphone to the same network that the demo device is connected to.
3. Connect a headphone using the 3.5-mm jack.
4. Open the Bonjour Browser application, and navigate to local -> World Wide Web HTML-over-HTTP -> xxxxxxxxxxxx@mysimplelink, as shown in Figure 18.

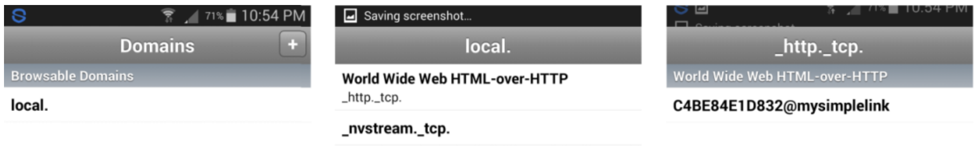
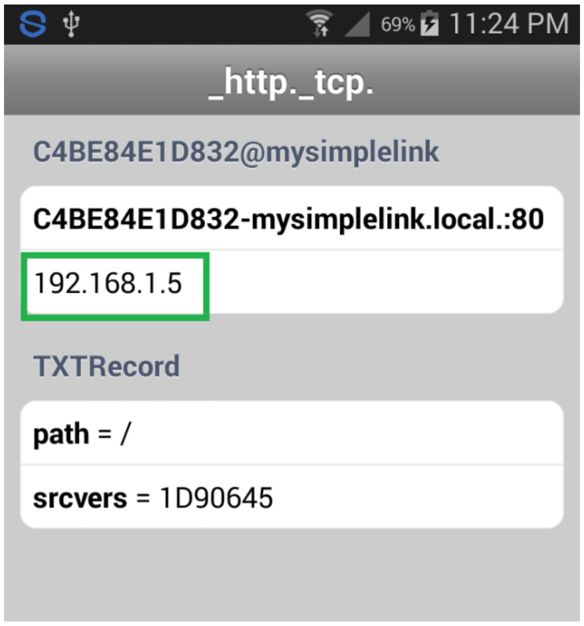


Figure 18. Bonjour Browser Application Navigation

5. Note down the IP address of the CC3200, as shown in Figure 19.



C4BE84E1D832@mysimplelink

C4BE84E1D832-mysimplelink.local.:80

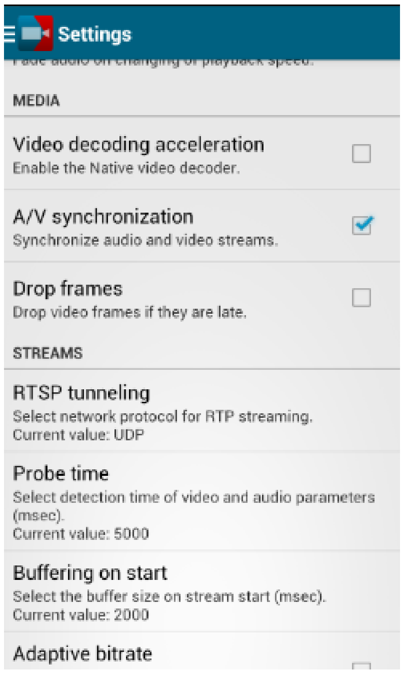
192.168.1.5

path = /

srcvers = 1D90645

Figure 19. IP Address Location in Bonjour Browser

6. Open the RTSP Player application and navigate to the Settings tab.
7. Uncheck Video decoding acceleration and Drop frames under MEDIA, as shown in Figure 20.



Fade audio on changing of playback speed.

Video decoding acceleration ☐
Enable the Native video decoder.

A/V synchronization ✓
Synchronize audio and video streams.

Drop frames ☐
Drop video frames if they are late.

RTSP tunneling
Select network protocol for RTP streaming.
Current value: UDP

Probe time
Select detection time of video and audio parameters (msec).
Current value: 5000

Buffering on start
Select the buffer size on stream start (msec).
Current value: 2000

Adaptive bitrate

Figure 20. Media Settings

8. Select UDP under STREAMS -> RTSP tunneling, as shown in Figure 21.

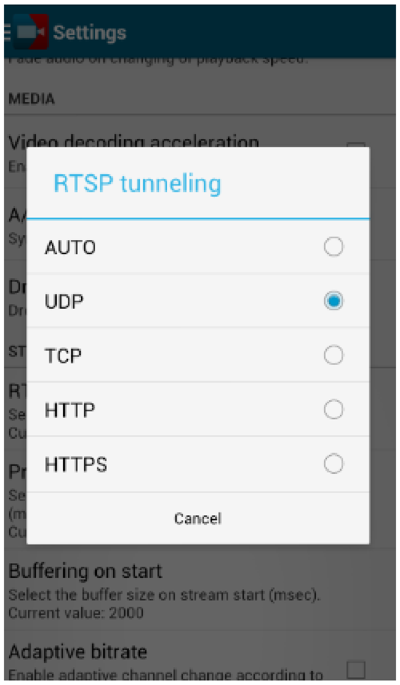

Settings
MEDIA
Video decoding acceleration
RTSP tunneling
AUTO
[✓ UDP]
TCP
HTTP
HTTPS
Cancel
Buffering on start
Select the buffer size on stream start (msec).
Current value: 2000
Adaptive bitrate
Enable adaptive channel change according to

Figure 21. RTSP Tunneling Settings

9. Enter 2000 under STREAMS -> Buffering on start, as shown in Figure 22.

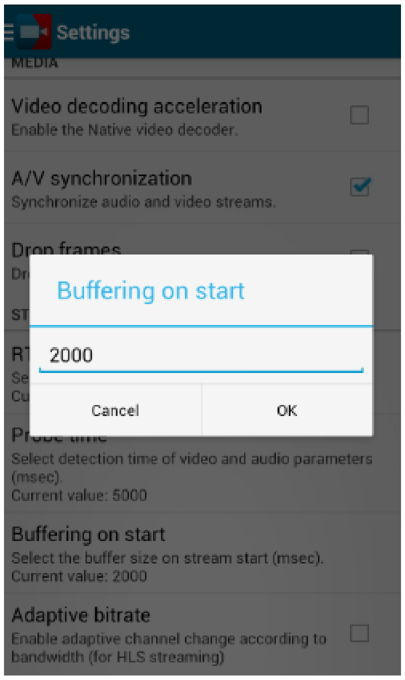
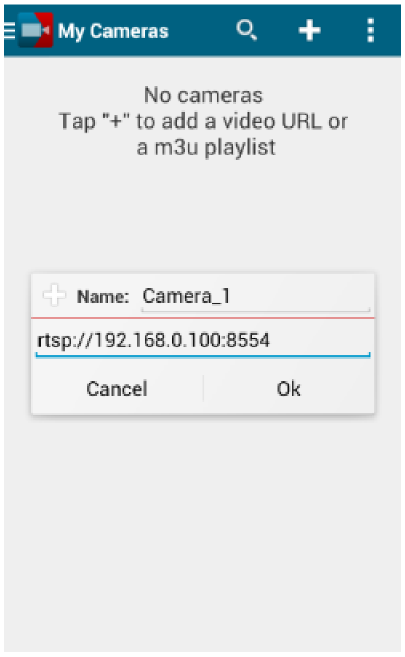

Settings
MEDIA
Video decoding acceleration
Enable the Native video decoder.
A/V synchronization
Synchronize audio and video streams.
Drop frames
Buffering on start
2000
Cancel
OK
Select detection time of video and audio parameters (msec).
Current value: 5000
Buffering on start
Select the buffer size on stream start (msec).
Current value: 2000
Adaptive bitrate
Enable adaptive channel change according to bandwidth (for HLS streaming)

Figure 22. STREAMS -> Buffering on Start Menu

10. Add a Camera by pressing on the “+” button as shown in Figure 23. Enter a Name and rtsp://<demo-device-ip>:8554, then press Ok.



No cameras
Tap "+" to add a video URL or a m3u playlist

Name: Camera_1

rtsp://192.168.0.100:8554

Figure 23. Adding a Video URL

11. The RTSP Player application connects to the device, and starts playing the live video and audio streams.

3 Limitations

- There is a latency of approximately 2 seconds in the streaming, mostly due to the buffering by the Smartphone application.
- The VLC application on Android does not process L16 payload, and thus cannot be used for audio-video streaming. Use RTSP Player instead (on Android smartphones) for simultaneous streaming of audio and video.
- The VLC application on iOS devices sometimes fails to play the audio streams beyond 15m, although the embedded application keeps streaming the data samples. This problem is not observed with RTSP Player on iOS devices.
- The sender reports, when sent to applications such as VLC, resulted in total audio silence/blackout. Thus, processing of SR/RR RTCP packets is not currently enabled. This might result in out-of-sync AV when the application is run over a prolonged period of time. Processing of these RTCP packets can be enabled by defining the macro RX_RR_TX_SR.
- The OV788 system is not power-optimized while in idle mode.

4 Design Files

To download the hardware design files for the CC3200-MODULE, see the design files at http://www.ti.com/tool/TIDC-CC3200-VIDEO.

5 Software Files

To download the software files for this reference design, see the design files at http://www.ti.com/tool/TIDC-CC3200-VIDEO.

6 Related Documentation

1. OV788 SIF Protocol Document

6.1 Trademarks

SimpleLink is a trademark of Texas Instruments.
ARM Cortex is a registered trademark of ARM Limited.
Apple, iPhone, iPad are registered trademarks of Apple.
Android is a trademark of Google, Inc..
Windows is a registered trademark of Microsoft.
OmniVision is a trademark of OmniVision Technologies, Inc..
Wi-Fi is a registered trademark of Wi-Fi Alliance.
All other trademarks are the property of their respective owners.

7 Appendices

7.1 RTSP Library APIs

- API: **rtsp_init**

  This function is used to initialize the RTSP Library. This function should be called before any other RTSP library function.

Arguments	p_param	Pointer to buffer containing the source description information
Return		0 on success, negative value on error

- API: **rtsp_deinit**

  Deinintilize the RTSP library.

Arguments	None	NA
Return		0 on success, negative value on error

- API: **rtsp_release_setup**

  Resets the setup-counter incremented on receiving an SETUP packet.

Arguments	None	NA
Return		0 on success, negative value on error

- API: **rtsp_packet_parser**

  This function processes the received RTSP packet, and creates the appropriate response packet.

Arguments	p_in_buffer	[Input] : Pointer to buffer containing RTSP packet
	rtp_pkt_size	[Input] : Size of the RTSP packet
	p_rtsp_session_config	[Input] : Pointer to structure (rtspModuleConfig) containing the information required to create RTSP response packet
	p_out_buffer	[Output] : Pointer to buffer for storing RTSP response packet
	rtsp_resp_pkt_len	[Input] : Output buffer size
	pkt_info	[Output] :Pointer to structure (rtspPacketInfo) containing the RTSP packet info
Return		Size of data in buffer, negative value on error

- Structure: **rtsp_session_config_s**

  This structure is used to pass the information required to create the RTSP response packet.

stream_port[MEDIA_STREAMS]	rtsp_stream_port_s	Socket port pair for a media stream
session_timeout	Long	Server session timeout value
session_timeout	Long	Server session number
timestamp	Unsigned long	Current timestamp value
multicast_ip	Unsigned char array	Server Multicast Destination IP
date	Unsigned char array	Date and Time string

- Structure: **rtsp_pkt_info_s**

  This structure is used to return the RTSP packet-related information.

pkt_type	rtsp_packet_type_e	RTSP packet type
pkt_response	rtsp_pkt_response_u	Packet payload information
seq_num	Long	RTSP request sequence number
setup_count	Long	Total Setup requests

- Structure: **rtsp_setup_data_s**

  This structure is used to return SETUP RTSP packet payload information.

rtp_port	Unsinged short	Client RTP port
rtcp_port	Unsinged short	Client RTCP port
is_multicast	char	Is the request for UDP multicast

7.2 RTP/RTCP Library APIs

- API: **rtp_init**

  Initializes the RTP library.

Arguments	None	NA
Return	Long	Size of data in output buffer, negative value on error

- API: **rtp_deinit**

  Deinitializes the RTP library.

Arguments	None	NA
Return	Long	Size of data in output buffer, negative value on error

- API: **rtp_process_rtcp_pkt**
  Creates the RTP for video p_rtp_profile.

Arguments	p_data	[Input] : Pointer to data buffer
	data_len	[Input] : Size of the data in buffer
	p_out_buffer	[Output] : Pointer to buffer for storing the output RTP packet
	out_buffer_len	[Input] : Maximum size of output buffer
	p_rtp_profile	[Input] : Pointer to structure (rtpProfile) containing the required profile and RTP header specific information to create the RTP packet
Return		Size of data in output buffer, negative value on error

- API: **rtp_create_sr_pkt**
  Creates the sender report packet.

Arguments	p_output	Pointer to buffer containing the output SR
	buffer_len	[Input] : Maximum size of the output buffer
	sr_info	[Input] : Pointer to structure (rtcpSRinfo) containing the information required to create SR packet
Return		Size of data in output buffer, negative error-code on error

- Structure: **rtp_profile_s**
  This structure is used to pass the RTP header, and profile specific information.

p_data	rtp_profile_data_s	RTP profile data
type	profile_type_e	Profile type. Currently only video is supported.
payload_format	payload_format_e	Payload format
timestamp	Unsigned long	Timestamp value
ssrc	Unsigned long	32-bit synchronization source identifier
seq_num	Unsigned short	RTP packet sequence number
payload_type	Unsigned char	Data payload type. This is the value used in the RTSP described response.