WL18xx TI Bluetooth Stack AUDM Demo App

Return to WL18xx AM335x TI Bluetooth Stack

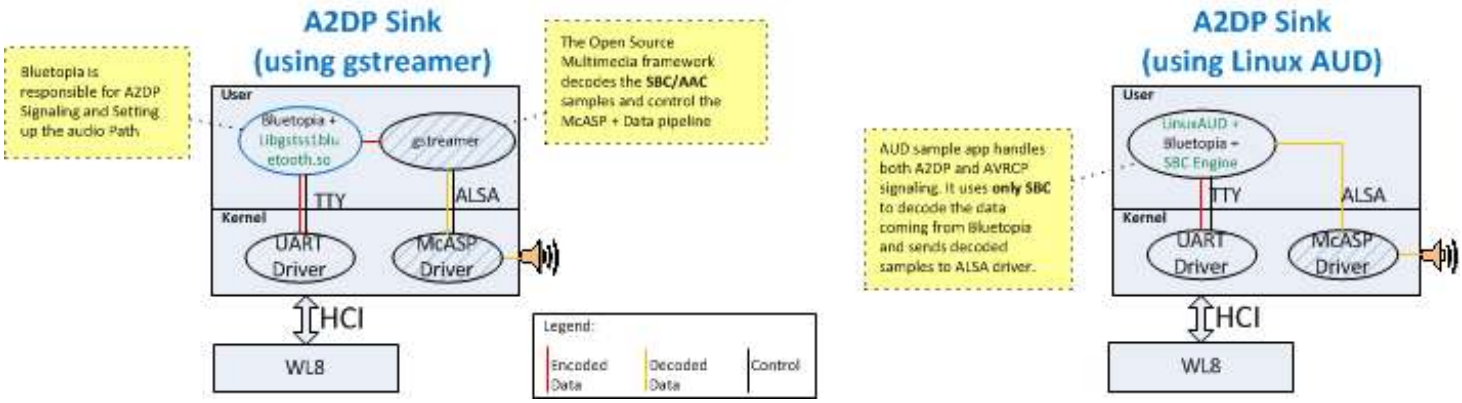
Contents

Demo Overview

Note: **: This instructions can be used to run this demo on the AM335x Platform.**

There are 2 ways to verify Audio use cases:
LinuxAUDM - is a sample application that operates on top of A2DP and AVRCP. It is compiled with SBC encoder/decoder and interfaces ALSA driver directly.
The Advanced Audio Distribution Profile (A2DP) Sink and Audio Video Remote Control Profile (AVRCP) allows a device to act as an Audio sink and can control and stream audio on an Audio source.
The LinuxAUDM application can connect to 5 audio sources as SINK. The application will always play the last played source stream.

GStreamer - is an open source multimedia framework which is connected to the Bluetopia stack through provided plugin. The gstreamer framewrok supports both AAC and SBC decoding.



Notes:
1. By default the BluetopiaPM is configured to support both AAC and SBC. This can cause an issue when using LinuxAUD for both Source and Sink (as the unsupported AAC decoding will be selected). To avoid this you can reset the value of BTPS_CONFIGURATION_AUD_SUPPORT_AAC in BTPSCFG.h and recompile the Module Configuration file (BTPMMODC) and the PM Server (SS1BTPM). Follow the instructions in "TI Bluetooth Stack Build Process".

2. To increase volume of a Sink device, you may use the following command:
"amixer cset name='PCM Playback Volume' 100%, 100%".
The percentage can be changed to control the absolute volume of the Headphone Out.

3. When the SINK is connected to several audio sources and one of the audio sources is an iOS device. The play from each device will take over any other stream, if exist, but if the stopped/paused device is an iOS device it will take 5 to 10 seconds for the iOS device to stop the stream. Pressing the play again before the iOS device stopped the stream will not work.

- It is recommended that the user visits the kit setup Getting Started Guide for AM335x (http://processors.wiki.ti.com/index.php/WL18xx_AM335x_TI_Bluetooth_Stack) pages before trying the application described on this page.

Running the Bluetooth Code

Once the am335x initialized on the terminal, follow the instructions;
Enter to the Sample Application folder, for this example “cd BluetopiaPM/bin”

```
root@am335x-evm:~# cd BluetopiaPM/bin/
root@am335x-evm:~/BluetopiaPM/bin#
```

Start the Server and the Demo application by typing “./SS1BTPM & ./LinuxAUDM”

```
root@am335x-evm:~/BluetopiaPM/bin# ./SS1BTPM & ./LinuxAUDM
[1] 1789
```

Now the help screen is displayed

```
**************************************************************************
* Command Options: 1) Initialize                                         *
*                  2) Cleanup                                            *
*                  3) QueryDebugZoneMask                                 *
*                  4) SetDebugZoneMask                                   *
*                  5) ShutdownService                                    *
*                  6) RegisterEventCallback,                             *
*                  7) UnRegisterEventCallback,                           *
*                  8) QueryDevicePower                                   *
*                  9) SetDevicePower                                     *
*                  10)QueryLocalDeviceProperties                         *
*                  11)SetLocalDeviceName                                 *
*                  12)SetLocalClassOfDevice                              *
*                  13)SetDiscoverable                                    *
*                  14)SetConnectable                                     *
*                  15)SetPairable                                        *
*                  16)StartDeviceDiscovery                               *
*                  17)StopDeviceDiscovery                                *
*                  18)QueryRemoteDeviceList                              *
*                  19)QueryRemoteDeviceProperties                        *
*                  20)AddRemoteDevice                                    *
*                  21)DeleteRemoteDevice                                 *
*                  22)DeleteRemoteDevices                                *
*                  23)PairWithRemoteDevice                               *
*                  24)CancelPairWithRemoteDevice                         *
*                  25)UnPairRemoteDevice                                 *
*                  26)QueryRemoteDeviceServices                          *
*                  27)RegisterAuthentication                             *
*                  28)UnRegisterAuthentication                           *
*                  29)PINCodeResponse                                    *
*                  30)PassKeyResponse                                    *
*                  31)UserConfirmationResponse                           *
*                  32)ChangeSimplePairingParameters                      *
*                  33)AUDRegisterAudioEndPoint                           *
*                  34)AUDUnRegisterAudioEndPoint                         *
*                  35)AUDRegisterRemoteControl                           *
*                  36)AUDUnRegisterRemoteControl                         *
*                  37)AUDConnectAudio                                    *
*                  38)AUDDisconnectAudio                                 *
*                  39)AUDConnectionRequestResponse                       *
*                  40)AUDChangeIncomingFlags                             *
*                  41)AUDPlayWAV                                         *
*                  42)AUDGetStreamStatus                                 *
*                  43)AUDGetConnectedSources                             *
*                  44)AUDQueryStreamState                                *
*                  45)AUDSetStreamState                                  *
*                  46)AUDQueryStreamFormat                               *
*                  47)AUDSetStreamFormat                                 *
*                  48)AUDQueryStreamConfiguration                        *
*                  49)AUDSendRemoteControlPassThroughCommand             *
*                  50)AUDConnectRemoteControl                            *
*                  51)AUDDisconnectRemoteControl                         *
*                  52)AUDQueryConnectedRemoteControls                    *
*                  53)AUDSendRemoteControlGetCapabilitiesCommand         *
*                  54)AUDSendRemoteControlSetAbsoluteVolumeCommand       *
*                  55)AUDSendRemoteControlRegisterNotificationCommand    *
*                  56)AUDSendRemoteControlRegisterNotificationResponse   *
*                  57)AUDSendRemoteControlGetTrackInformationCommand     *
*                  58)AUDSendRemoteControlGetPlayStatusCommand           *
*                  59)AUDChangeSupportedSBCCodecBitpoolValues            *
*                  60)SetSniff                                           *
*                  61)EnableBluetoothDebug                               *
*                  Help, Quit.                                           *
**************************************************************************
```

Demo Application

This section provides a description of how to use the demo application to connect an audio source to it and communicate over Bluetooth.

Audio Profile Manager (AUDM) - A2DP and AVRCP

A2DP Source and AVRCP Target Roles

To verify the AUDM A2DP source and AVRCP target role functionality, after running the "LinuxAUDM" executable, initializing, and powering on the local device if necessary, first register an audio endpoint as a source.

a) Initialize the profile by typing “1 1” or “Initialize 1”.
b) Powering up the Bluetooth device by typing “9 1” or "SetDevicePower 1”, if the device is already powered up you’ll receive an error that the device is already powered up

```
AUDM>Initialize  1

BTPM_Initialize() Success: 0.
DEVM_RegisterEventCallback() Success: 5.
killall: gst-launch-1.0: no process killed
AUDM>
AUDM>
AUDM>SetDevicePower  1

echo 117 > /sys/class/gpio/export
sh: line 0: echo: write error: Device or resource busy
echo out > /sys/class/gpio/gpio117/direction
echo 0 > /sys/class/gpio/gpio117/value
echo 1 > /sys/class/gpio/gpio117/value
echo 0 > /sys/class/gpio/gpio117/value
echo 1 > /sys/class/gpio/gpio117/value
BT COMM PORT (/dev/ttyS1): 1
Changing HCI baud rate to 3000000
Status: Executing BTS Script /lib/firmware/TIInit_6.7.16.bts.
Status: BTS Script successfully executed.
DEVM_PowerOnDevice() Success: 0.
AUDM>
Device Powered On.
AUDM>
```

c) Registering an audio endpoint as a source

```
Usage: AUDRegisterAudioEndPoint [Local Stream Type (0 = Source, 1 = Sink)].
```

```
AUDM>AUDRegisterAudioEndPoint  0

AUDM_Register_Data_Event_Callback() Success: 1.
Audio Source Callback Registered.
AUDM>
```

d) Connect to a remote device registered as a sink.

```
Usage: AUDConnectAudio [Local Stream Type (0 = Source, 1 = Sink)] [BD_ADDR] [Flags (Optional - 1:Authentication, 2:Encryption)].
```

```
AUDM>AUDConnectAudio 0 0017E95066BC

AUDM_Connect_Audio_Stream() Success: 0.
AUDM>
Remote Device Properties Changed.
BD_ADDR:       0017E95066BC
Device Flags:  0x800000CD
Connect State: TRUE
AUDM>
Remote Device Properties Changed.
BD_ADDR:       0017E95066BC
Device Flags:  0x800000DD
Encrypt State: TRUE
AUDM>
LinuxAUDM: Audio Stream Connection Status: 0, Success.
    Type     : Source
    Address  : 0017E95066BC
    Index  : -1
    MTU      : 672
    Frequency: 44100
    Channels : 2
    Flags    : 0
AUDM>
LinuxAUDM: Audio Stream Connected.
    Type     : Source
    Address  : 0017E95066BC
    MTU      : 672
    Frequency: 44100
    Channels : 2
    Flags    : 0

LinuxAUDM: Stream number 1 connected.
   Result:          0
   Response Length: 4
   Response Data:   00010000
LinuxAUDM: The device role is Master
AUDM>
LinuxAUDM: Remote Control Connected.
    Address: 0017E95066BC

LinuxAUDM: Set sniff mode, Success: 0.
AUDM>
Remote Device Properties Changed.
BD_ADDR:       0017E95066BC
Device Flags:  0x800000FD
Sniff State : TRUE (625 ms)
AUDM>
```

Note - If remote device disconnects right after connection, use the **RegisterAuthentication** *command and try to connect again.*

e) Play a .wav file.

```
Usage: AUDPlayWAV [BD_ADDR] [WAV File Name] [Restart/Resume - 0/1 (Optional)].
```

```
AUDM>AUDPlayWAV 0017E95066BC wav2.wav

Play wav file name= wav2.wav
Parsed WAV File:
    AudioDataOffset:       44
    AudioDataLength:       9580500
    Format:                1
    Channels:              2
    SamplesPerSecond:      44100
    AverageBytesPerSecond: 176400
    BlockSize:             4
    BitsPerSample:         16
    ValidBitsPerSample:    16


LinuxAUDM: Set Active mode, Success: 0.
Starting Playback Thread.
AUDM>:: StreamConfig.MediaCodecType = 0(0)
:: StreamConfig.MediaCodecInfoLength = 4(4)
Building SBC configuration based on these connection parameters:
    MediaMTU:     672
    Frequency:    44100
    Channels:     2
    FormatFlags:  0x00
    CodecType:    SBC
    SFCM:         0x21
    BLSAM:        0x15
    Min Bit Pool: 10
    Max Bit Pool: 80

Initializing SBC encoder using parameters:
    Freq:         44100
    Block Size:   16
    Channel Mode: Joint Stereo
    Allocation:   Loudness
    Subbands:     8
    Bit Pool:     55

Timing Parameters:
    MTU:                671
    Frame Size:         123
    Frames per Packet:  5
    Packet Delay (NS):  14512471
    Packet Delay:       0.014512000 sec
    Extra Time/Pkt:     471
    Underrun Time:      999529

Total audio data stream size: 9580500
Total read: 102400

Remote Device Properties Changed.
BD_ADDR:       84DD2069DEBC
Device Flags:  0x800000DD
Sniff State : FALSE
AUDM>
LinuxAUDM: Change Audio Stream State Status: Success
    Type : Source
    State: Playing
    Address  : 84DD2069DEBC
    Index  : 0
LinuxAUDM: Remote device index = 0 start streaming
AUDM>Total read: 204800
Sent SBC packet # 100 (500 total frames)
Total read: 307200
Total read: 409600
Total read: 512000
Sent SBC packet # 200 (1000 total frames)
Total read: 614400
```

f) Register as an AVRCP target.

```
Usage: AUDRegisterRemoteControl [Service Type (0 = Target, 1 = Controller)].
```

```
AUDM>AUDRegisterRemoteControl 0

AUDM_Register_Remote_Control_Event_Callback() Success: 14.
AUDM>
```

g) Send a remote control command from the remote device. The local device should then receive a 'Remote Control Command Indication'.

```
AUDM>
AUDM: Remote Control Command Indication.
    Address      : 001BDC0FC9DC
    TransactionID: 3
    Message Type :  PassThrough
       Command Type: 0
       Subunit Type: 9
       Subunit ID  : 0
       Operation ID: 68
       State Flag  : FALSE
       Data Length : 0
AUDM>
AUDM: Remote Control Command Indication.
    Address      : 001BDC0FC9DC
    TransactionID: 3
    Message Type :  PassThrough
       Command Type: 0
       Subunit Type: 9
       Subunit ID  : 0
       Operation ID: 68
       State Flag  : FALSE
       Data Length : 0
AUDM>
AUDM: Remote Control Command Indication.
    Address      : 001BDC0FC9DC
```

```
    TransactionID: 4
    Message Type :  PassThrough
       Command Type: 0
       Subunit Type: 9
       Subunit ID  : 0
       Operation ID: 68
       State Flag  : TRUE
       Data Length : 0
AUDM>
AUDM: Remote Control Command Indication.
    Address      : 001BDC0FC9DC
    TransactionID: 4
    Message Type :  PassThrough
       Command Type: 0
       Subunit Type: 9
       Subunit ID  : 0
       Operation ID: 68
       State Flag  : TRUE
       Data Length : 0
AUDM>
```

A2DP Sink and AVRCP Controller Roles

To verify the AUDM A2DP Sink and AVRCP Controller role functionality, after running the "LinuxAUDM" executable, initializing, and powering on the local device if necessary, first register an audio endpoint as a sink.

a) Initialize the profile by typing “1 1” or “Initialize 1”.
b) Powering up the Bluetooth device by typing “9 1” or "SetDevicePower 1”, if the device is already powered up you’ll receive an error that the device is already powered up

```
AUDM>Initialize 1

BTPM_Initialize() Success: 0.
DEVM_RegisterEventCallback() Success: 5.
AUDM>SetDevicePower 1

echo 44 > /sys/class/gpio/export
sh: line 0: echo: write error: Device or resource busy
echo out > /sys/class/gpio/gpio44/direction
echo 0 > /sys/class/gpio/gpio44/value
echo 1 > /sys/class/gpio/gpio44/value
echo 0 > /sys/class/gpio/gpio44/value
echo 1 > /sys/class/gpio/gpio44/value
BT COMM PORT (/dev/ttyS4): 1
Changing HCI baud rate to 3000000
Status: Executing BTS Script /lib/firmware/TIInit_11.8.32.bts.
Status: BTS Script successfully executed.
DEVM_PowerOnDevice() Success: 0.
AUDM>
Device Powered On.
AUDM>
```

c) Registering an audio endpoint as a source

```
Usage: AUDRegisterAudioEndPoint [Local Stream Type (0 = Source, 1 = Sink)].
```

```
AUDM>AUDRegisterAudioEndPoint 1

AUDM_Register_Data_Event_Callback() Success: 1.
Audio Sink Callback Registered.
AUDM>
```

d) Connect to a remote device registered as a source.

```
Usage: AUDConnectAudio [Local Stream Type (0 = Source, 1 = Sink)] [BD_ADDR] [Flags (Optional - 1:Authentication, 2:Encryption)].
```

```
AUDM>AUDConnectAudio 1 40B0FA0D77BC

AUDM_Connect_Audio_Stream() Success: 0.
AUDM>
Remote Device Properties Changed.
BD_ADDR:       40B0FA0D77BC
Device Flags:  0x800000CD
Connect State: TRUE
AUDM>
Remote Device Properties Changed.
BD_ADDR:       40B0FA0D77BC
Device Flags:  0x800000DD
Encrypt State: TRUE
AUDM>
LinuxAUDM: Audio Stream Connection Status: 0, Success.
    Type     : Sink
    Address  : F895C7763366
    Index  : -1
    MTU      : 1008
    Frequency: 44100
    Channels : 2
    Flags    : 0
AUDM>
LinuxAUDM: Audio Stream Connected.
    Type     : Sink
    Address  : F895C7763366
    MTU      : 1008
    Frequency: 44100
    Channels : 2
```

```
    Flags    : 0

LinuxAUDM: Stream number 1 connected.
   Result:          0
   Response Length: 4
   Response Data:   00010001
LinuxAUDM: The device role is Slave

LinuxAUDM: DEVM_SendRawHCICommand() Switch Role HCI Link Policy Command. Success: 0.
AUDM>
AUDM>
LinuxAUDM: Remote Control Connected.
    Address: F895C7763366

LinuxAUDM: Set sniff mode, Success: 0.
AUDM>
Remote Device Properties Changed.
BD_ADDR:       F895C7763366
Device Flags:  0x800000FD
Sniff State : TRUE (625 ms)
AUDM>
```

e) Play music on the remote device and make sure the local device is receiving the audio data.

```
AUDM>
Remote Device Properties Changed.
BD_ADDR:       F895C7763366
Device Flags:  0x800000DD
Sniff State : FALSE
AUDM>
LinuxAUDM: Audio Stream State Changed.
    Type   : Sink
    State  : Playing
    Address: F895C7763366
    Index  : 0
LinuxAUDM: Remote device index = 0 is streaming now

LinuxAUDM: DEVM_SendRawHCICommand() Link Policy - Flow Specification. Success: 0.

PM_VS_DDIPCommand() DDIP Success: 0.
```

f) Register as a remote control controller, and then connect to the remote device.

```
Usage: AUDRegisterRemoteControl [Service Type (0 = Target, 1 = Controller)].
```

```
AUDM>AUDRegisterRemoteControl 1

AUDM_Register_Remote_Control_Event_Callback() Success: 2.
AUDM>
Remote Device Properties Changed.
BD_ADDR:       40B0FA0D77BC
Device Flags:  0x8000008D
Connect State: TRUE
AUDM>
Remote Device Properties Changed.
BD_ADDR:       40B0FA0D77BC
Device Flags:  0x8000009D
Encrypt State: TRUE
AUDM>
AUDM: Audio Stream Connected.
    Type     : Sink
    Address  : 40B0FA0D77BC
    MTU      : 668
    Frequency: 44100
    Channels : 2
    Flags    : 0
AUDM>
AUDM: Audio Stream Connected.
    Type     : Sink
    Address  : 40B0FA0D77BC
    MTU      : 668
    Frequency: 44100
    Channels : 2
    Flags    : 0
AUDM>
AUDM: Remote Control Connected.
    Address: 40B0FA0D77BC
AUDM>
AUDM: Remote Control Connected.
    Address: 40B0FA0D77BC
AUDM>
Remote Device Properties Changed.
BD_ADDR:       40B0FA0D77BC
Device Flags:  0x8000008D
Sniff State  : TRUE (500 ms)
AUDM>
Remote Device Properties Changed.
BD_ADDR:       40B0FA0D77BC
Device Flags:  0x8000009D
Sniff State  : FALSE
AUDM>
Remote Device Properties Changed.
BD_ADDR:       40B0FA0D77BC
Device Flags:  0x8000008D
Sniff State  : TRUE (500 ms)
AUDM>
```

g) Play music on the remote device and from the local device send a remote control command. The command's third parameter is the remote control command to send, 0 = pause, 1 = play, 2 = stop, 3 = vol. up, 4 = vol. down, and 5 = specify command (takes an additional specify command value).

```
Usage: AUDSendRemoteControlPassThroughCommand [BD_ADDR] [Timeout (ms)] [Passthrough Command (0 = Pause, 1 = Play, 2 = Stop, 3 = Vol. Up, 4 = Vol. Down,
5 = Specify Command)] [Specify Command Value (only used with Specify Command)].
```

```
AUDM>AUDSendRemoteControlPassThroughCommand 40B0FA0D77BC 0 3000

Pause Pass-through Command specified.
Remote Control Command Sent: PRESSED, Transaction ID = 146337328.
Remote Control Command Sent: RELEASED, Transaction ID = 146337328.
AUDM>

AUDM: Audio Stream State Changed.
    Type : Sink
    State: Stopped
Waiting for thread...
Exit Decode Thread
Wait done
AUDM>
AUDM: Audio Stream State Changed.
    Type : Sink
    State: Stopped
AUDM>
Remote Device Properties Changed.
BD_ADDR:       40B0FA0D77BC
Device Flags:  0x800000BD
Sniff State  : TRUE (500 ms)
AUDM>
```

A2DP Sink and AVRCP Controller for multiple Audio sources

The BluetopiaPM layer can handle multiple audio sources, as A2DP sink and AVRCP controller. The LinuxAUDM application can demonstrate connection/streaming/controlling with multiple sources. To demonstrate the feature, start the application and connect to the device, running the LinuxAUDM, with several phones. The phones will act as an audio sources and AVRCP target. The application can connect to maximum 5 sources and play the audio. The application will try to be the master of each connected source and enter to sniff mode with the remote device. When one of the sources will start streaming, the application will send AVRCP pause to any other playing source and start stream the audio data that comming from the last playing source. When the streaming starts, the device will move to active mode and the Bluetooth ACL link will move to higher priority over the other connections and over the scans.

- When using iOS device as source and pausing the stream, the user must wait to the stream stopped event, that appear 5-10 seconds after the pausing, before pressing play again.

a) Initialize the LinuxAUDM application by typing “1 1” or “Initialize 1”.
b) Powering up the Bluetooth device by typing “9 1” or "SetDevicePower 1”, if the device is already powered up you’ll receive an error that the device is already powered up.

```
AUDM>Initialize 1

BTPM_Initialize() Success: 0.
DEVM_RegisterEventCallback() Success: 5.
AUDM>SetDevicePower 1

echo 44 > /sys/class/gpio/export
sh: line 0: echo: write error: Device or resource busy
echo out > /sys/class/gpio/gpio44/direction
echo 0 > /sys/class/gpio/gpio44/value
echo 1 > /sys/class/gpio/gpio44/value
echo 0 > /sys/class/gpio/gpio44/value
echo 1 > /sys/class/gpio/gpio44/value
BT COMM PORT (/dev/ttyS4): 1
Changing HCI baud rate to 3000000
Status: Executing BTS Script /lib/firmware/TIInit_11.8.32.bts.
Status: BTS Script successfully executed.
DEVM_PowerOnDevice() Success: 0.
AUDM>
Device Powered On.
AUDM>
```

c) Registering an audio endpoint as a SINK.

```
Usage: AUDRegisterAudioEndPoint [Local Stream Type (0 = Source, 1 = Sink)].
```

```
AUDM>AUDRegisterAudioEndPoint 1

AUDM_Register_Data_Event_Callback() Success: 1.
Audio Sink Callback Registered.
AUDM>
```

d) Register as a remote control controller.

```
Usage: AUDRegisterRemoteControl [Service Type (0 = Target, 1 = Controller)].
```

```
AUDM>AUDRegisterRemoteControl 1
AUDM_Register_Remote_Control_Event_Callback() Success: 2.
AUDM>
```

e) If the remote audio source is not paired, register Authentication with 27 or use the RegisterAuthentication command.

When the remote audio source will try to connect, a message will appear on the terminal asks to press UserConfirmationResponse with 1.

```
AUDM>RegisterAuthentication
DEVM_RegisterAuthentication() Success: 5.
```

```
AUDM>
```

f) Connect with the audio sources to the sink. In the prints below two remote audio sources connected to the sink device.

After the connection the application also prints the stream number includes the role and the mode change to sniff.

```
AUDM>
Remote Device Properties Changed.
BD_ADDR:       E0B52D1C6A9C
Device Flags:  0x8000008D
Connect State: TRUE
AUDM>
Remote Device Properties Changed.
BD_ADDR:       E0B52D1C6A9C
Device Flags:  0x8000009D
Encrypt State: TRUE
AUDM>
LinuxAUDM: Audio Stream Connected.
    Type     : Sink
    Address  : E0B52D1C6A9C
    MTU      : 672
    Frequency: 44100
    Channels : 2
    Flags    : 0

LinuxAUDM: Stream number 1 connected.
   Result:          0
   Response Length: 4
   Response Data:   00010001
LinuxAUDM: The device role is Slave

LinuxAUDM: DEVM_SendRawHCICommand() Switch Role HCI Link Policy Command. Success: 0.
AUDM>
LinuxAUDM: Remote Control Connected.
    Address: E0B52D1C6A9C

LinuxAUDM: Set sniff mode, Success: 0.
AUDM>
Remote Device Properties Changed.
BD_ADDR:       E0B52D1C6A9C
Device Flags:  0x800000BD
Sniff State : TRUE (625 ms)
AUDM>
Remote Device Properties Changed.
BD_ADDR:       E0B52D1C6A9C
Device Flags:  0x8000009D
Sniff State : FALSE
AUDM>
AUDM>
AUDM>
Remote Device Properties Changed.
BD_ADDR:       E0B52D1C6A9C
Device Flags:  0x800000BD
Sniff State : TRUE (240 ms)
AUDM>
Remote Device Properties Changed.
BD_ADDR:       84DD2069DEBC
COD:           0x000000
AUDM>
Remote Device Properties Changed.
BD_ADDR:       84DD2069DEBC
Device Flags:  0x8000008D
Connect State: TRUE
AUDM>
Remote Device Properties Changed.
BD_ADDR:       84DD2069DEBC
Device Flags:  0x8000009D
Encrypt State: TRUE
AUDM>
LinuxAUDM: Audio Stream Connected.
    Type     : Sink
    Address  : 84DD2069DEBC
    MTU      : 672
    Frequency: 44100
    Channels : 2
    Flags    : 0

LinuxAUDM: Stream number 2 connected.
   Result:          0
   Response Length: 4
   Response Data:   00020001
LinuxAUDM: The device role is Slave

LinuxAUDM: DEVM_SendRawHCICommand() Switch Role HCI Link Policy Command. Success: 0.
AUDM>
LinuxAUDM: Remote Control Connected.
    Address: 84DD2069DEBC

LinuxAUDM: Set sniff mode, Success: 0.
AUDM>
Remote Device Properties Changed.
BD_ADDR:       84DD2069DEBC
Device Flags:  0x800000BD
Sniff State : TRUE (480 ms)
AUDM>
```

g) After the 2 sources connected with A2DP and AVRCP, the prints below shows an example usage of the following scenario:

- Press PLAY from one audio source. The device will start streaming the music of the playing source.
- Press PLAY from the second source. The device will start streaming the music of the second playing source and send PAUSE to the first one that played.

```
AUDM>
Remote Device Properties Changed.
BD_ADDR:       E0B52D1C6A9C
Device Flags:  0x8000009D
Sniff State : FALSE
```

```
AUDM>
LinuxAUDM: Audio Stream State Changed.
    Type   : Sink
    State  : Playing
    Address: E0B52D1C6A9C
    Index  : 0
LinuxAUDM: Remote device index = 0 is streaming now

LinuxAUDM: DEVM_SendRawHCICommand() Link Policy - Flow Specification. Success: 0.

PM_VS_DDIPCommand() DDIP Success: 0.
AUDM>
LinuxAUDM: Audio Stream State Changed.
    Type   : Sink
    State  : Playing
    Address: 84DD2069DEBC
    Index  : 1
LinuxAUDM: Remote device index 0 is playing now, stop the stream first
LinuxAUDM: Send AUDM_Send_Remote_Control_Command PAUSE
LinuxAUDM: AUDSendRemoteControlPassThroughCommand function returned- 1

LinuxAUDM: DEVM_SendRawHCICommand() Link Policy - Flow Specification. Success: 0.

Remote Device Properties Changed.
BD_ADDR:       84DD2069DEBC
Device Flags:  0x8000009D
Sniff State : FALSE
AUDM>
LinuxAUDM: Set sniff mode, Success: 0.
LinuxAUDM: Clean the decoder
LinuxAUDM: Start the decoder with next playing remote device.

Remote Device Properties Changed.
BD_ADDR:       E0B52D1C6A9C
Device Flags:  0x800000BD
Sniff State : TRUE (625 ms)
AUDM>LinuxAUDM: Remote device index = 1 is streaming now

LinuxAUDM: DEVM_SendRawHCICommand() Link Policy - Flow Specification. Success: 0.

PM_VS_DDIPCommand() DDIP Success: 0.
AUDM>
Remote Device Properties Changed.
BD_ADDR:       E0B52D1C6A9C
Device Flags:  0x8000009D
Sniff State : FALSE
AUDM>
LinuxAUDM: Remote Control Command Confirmation.
    Address      : E0B52D1C6A9C
    TransactionID: 1
    Status       : 0 (Success)
    Message Type : PassThrough
       Response Code: 0x09 (Accepted)
       Subunit Type : 9
       Subunit ID   : 0
       Operation ID : 70
       State Flag   : FALSE
       Data Length  : 0
AUDM>
LinuxAUDM: Audio Stream State Changed.
    Type   : Sink
    State  : Stopped
    Address: E0B52D1C6A9C
    Index  : 0
LinuxAUDM: Remote device index = 0 stopped playing
LinuxAUDM: Remote device index = 1 is playing
AUDM>
```

h) Send from the SINK, PAUSE Pass Through command over AVRCP as a controller, to pause the stream. The second source will pause the stream and no stream is heard on the SINK.

```
Usage: AUDSendRemoteControlPassThroughCommand [BD_ADDR] [Timeout (ms)] [Passthrough Command (0 = Pause, 1 = Play, 2 = Stop, 3 = Vol. Up, 4 = Vol. Down,
5 = Specify Command)] [Specify Command Value (only used with Specify Command)].
```

```
AUDM>AUDSendRemoteControlPassThroughCommand  84DD2069DEBC 5000 0

Pause Pass-through Command specified.
Fill in the Pass Through Command dataSend AUDM_Send_Remote_Control_CommandRemote Control Command Sent: PRESSED, Transaction ID = 2.
Remote Control Command Sent: RELEASED, Transaction ID = 3.
AUDM>
LinuxAUDM: Audio Stream State Changed.
    Type   : Sink
    State  : Stopped
    Address: 84DD2069DEBC
    Index  : 1
LinuxAUDM: Remote device index = 1 stopped playing
LinuxAUDM: No remote is playing now

LinuxAUDM: DEVM_SendRawHCICommand() Link Policy - Flow Specification. Success: 0.

LinuxAUDM: Set sniff mode, Success: 0.
AUDM>
LinuxAUDM: Remote Control Command Confirmation.
    Address      : 84DD2069DEBC
    TransactionID: 2
    Status       : 0 (Success)
    Message Type : PassThrough
       Response Code: 0x09 (Accepted)
       Subunit Type : 9
       Subunit ID   : 0
       Operation ID : 70
       State Flag   : FALSE
       Data Length  : 0
AUDM>
LinuxAUDM: Remote Control Command Confirmation.
    Address      : 84DD2069DEBC
    TransactionID: 3
    Status       : 0 (Success)
```

```
   Message Type : PassThrough
      Response Code: 0x09 (Accepted)
      Subunit Type : 9
      Subunit ID   : 0
      Operation ID : 70
      State Flag   : TRUE
      Data Length  : 0
AUDM>
Remote Device Properties Changed.
BD_ADDR:       84DD2069DEBC
Device Flags:  0x800000BD
Sniff State : TRUE (625 ms)
```

NOTE: Before start streaming, the device makes the following actions: 1. Change state from sniff to active mode. 2. Sets higher priority to the streaming Bluetooth link over other links by sending Link Policy, Flow Specification command. 3. Sets higher priority to the streaming Bluetooth link over scans (inquiry and page).
When the stream is stopped, the device will return to the normal priority over other Bluetooth links and over scans and will enter sniff mode with the stopped device.

GStreamer

A GStreamer plugin is provided with the Bluetopia Platform Manager Distribution.
NOTE: The plugin needs to be copied to the target's root file system by using the commands on GStreamer Plugin.

The distribution you have been provided supports SBC and AAC A2DP Capabilities Negotiation. There is no requirement to use GStreamer when using A2DP Source or A2DP Sink with SBC or AAC, but it can be used optionally if GStreamer is the preferred media framework.

As with all demos, the SS1BTPM server must be running in the background. It may also be required to have a demo running which can handle authentication events and to also set the Class of Device attribute of the Blueooth Module. In these GStreamer examples we will use LinuxDEVM to handle authentication and to set the Class of Device. First start the LinuxDEVM and sample and then run the following:

```
DEVM>Initialize 1
DEVM>SetDevicePower 1
DEVM>SetLocalClassOfDevice 0x040424
DEVM>SetDiscoverable 1
DEVM>SetConnectable 1
DEVM>SetPairable 1
DEVM>RegisterAuthentication
DEVM>PairWithRemoteDevice <bluetooth_device_address>
```

A2DP Sink

SBC

The following GStreamer pipeline can be used to receive and playback encoded SBC Audio:

1) Press "aplay -L | grep :CARD" in order to check which sound cards are installed.

```
# aplay -L | grep :CARD
default:CARD=WILINK8BT
sysdefault:CARD=WILINK8BT
default:CARD=AM335xEVM
sysdefault:CARD=AM335xEVM
```

if you have one default and one sysdefault cards use:

```
# gst-launch-1.0 ss1a2dpsrc deccaps=sbcdeccaps ! ss1rtpsbcdepay ! capsfilter name=sbcdeccaps ! ss1sbcdec ! alsasink
```

if you have two default and two sysdefault cards, as in the example use:

```
# gst-launch-1.0 ss1a2dpsrc deccaps=sbcdeccaps ! ss1rtpsbcdepay ! capsfilter name=sbcdeccaps ! ss1sbcdec ! alsasink device="hw:1"
```

Note: You'll need to change the device HW number to the default:card that you want to use. (In this example, the second card - AM335xEVM).

Note: The "deccaps" attribute is used to specify the name of a capsfilter element that has been placed before the SBC Decoder element. This is required to correctly handle capabilities negotiation in GStreamer and to specify the capabilities of the A2DP Stream to the SBC Decoder.

Note: If your local device is configured to support AAC and the remote device also supports AAC then this pipeline will not work. In this scenario Bluetopia PM will use the highest quality codec available, which would be AAC, and SBC audio will not be sent over the air. The following section describes how to decode and playback AAC using GStreamer.

AAC

Note: AAC is disabled in the pre-built executables provided with the distribution. To enable AAC set the AAC_SUPPORT_ENABLED constant in BluetopiaPM/btpmmodc/server/BTPMMODC.c to 1 and rebuild SS1BTPM using the instructions in the TI Bluetooth Stack for WL18xx - Build Process.

The following GStreamer pipeline can be used to receive and playback encoded AAC Audio:

```
# gst-launch-1.0 ss1a2dpsrc deccaps=aacdeccaps ! ss1rtplatmdepay checkmuxfragments=false ! capsfilter name=aacdeccaps ! faad ! alsasink
```

Note: The "deccaps" attribute is required to correctly handle capabilities negotiation in GStreamer and to specify the capabilities of the A2DP Stream to the AAC Decoder.

A2DP Source

SBC

The following GStreamer pipeline can be used to encode and send SBC Audio:

```
# gst-launch-1.0 audiotestsrc ! audioresample ! ss1sbcenc ! capsfilter name=sbcenccaps ! ss1rtpsbcpay mtu=800
! ss1a2dpsink deviceaddress=<bluetooth_device_address> enccaps=sbcenccaps
```

Note: The "enccaps" attribute of ss1a2dpsink is used to specify the name of a capsfilter element that has been placed after the SBC Encoder element. This is required to correctly handle capabilities negotiation in GStreamer and to specify the capabilities of the A2DP Stream to the SBC Encoder.

Note: You must be on a device that does not support the AAC codec for this pipeline to work. If the remote device supports the AAC codec, then Bluetopia PM will use the AAC codec, and in this case the following section describes how to encode and send AAC.

AAC

When AAC is enabled with the AAC_SUPPORT_ENABLED constant in /opt/ss1/BluetopiaPM/btpmmodc/server/BTPMMODC.c it will enable the most commonly used AAC A2DP configuration, which is with MPEG version 2 support, MPEG version 4 is not enabled with the AAC_SUPPORT_ENABLED constant.

In the TI SDK the avenc_aac GStreamer AAC encoder element is included, however, this element only includes support for MPEG 4. The faac encoder element, which is part of GStreamer's gst-plugins-bad package, includes support for MPEG 2, however, this element is not included in TI's SDK.

To send encoded AAC audio as an A2DP source the user will have to either:

1. Cross-compile the faac element to add MPEG 2 encoding support to GStreamer
2. Re-build the PM Server (SS1BTPM) to include MPEG 4 support

To re-build the PM Server to add MPEG 4 support modify /opt/ss1/BluetopiaPM/btpmmodc/server/BTPMMODC.c and add one or more of the MPEG 4 Flags (e.g. AUD_STREAM_FORMAT_FLAGS_AAC_SUPPORT_MPEG4_LC) to the AAC Stream Formats in the SRCInitializationInfo structure. Then re-build the PM Server (SS1BTPM). For instructions on re-building the PM server please refer to the TI Bluetooth Stack for WL18xx - Build Process.

In this example we'll use the faac AAC encoder element to demonstrate encoding and sending AAC MPEG 2 audio, which is the most common AAC format used with A2DP. The following GStreamer pipeline can be used to encode and send AAC Audio:

```
# gst-launch-1.0 audiotestsrc ! audioresample ! faac ! capsfilter name=aacenccaps ! ss1rtplatmpay mtu=800
! ss1a2dpsink deviceaddress=<bluetooth_device_address> enccaps=aacenccaps
```

Note: The "enccaps" attribute is required to correctly handle capabilities negotiation in GStreamer and to specify the capabilities of the A2DP Stream to the AAC encoder.

Troubleshooting

If you are having issues with getting a GStreamer pipeline to work to the -v and --gst-debug-level flags can be used to help determine the problem. The -v flag tells gst-launch to dump additional information to the terminal, for example the negotiated capabilities. And the --gst-debug-level option is used to specify what level of debugging information to dump to the terminal. Stonestreet One's GStreamer plugin outputs GStreamer Error, Warning, and Info error messages.

For example, to enable the Error and Warning GStreamer messages as an A2DP Sink the following pipeline can be used:

```
# gst-launch-1.0 -v --gst-debug-level=2 ss1a2dpsrc deccaps=sbcdeccaps ! ss1rtpsbcdepay ! capsfilter name=sbcdeccaps ! ss1sbcdec ! alsasink
```

{{

1. switchcategory:MultiCore=
   - For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
   - For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **WL18xx TI Bluetooth Stack AUDM Demo App** here.

Keystone=

- For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
- For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **WL18xx TI**

C2000=*For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article* ***WL18xx TI Bluetooth Stack AUDM***

DaVinci=*For technical support on DaVincoplease post your questions on The DaVinci Forum. Please post only comments about the article* ***WL18xx TI Bluetooth Stack AUDM Demo App*** *here.*

MSP430=*For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article* ***WL18xx TI Bluetooth Stack AUDM Demo App*** *here.*

OMAP35x=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article* ***WL18xx TI Bluetooth Stack AUDM Demo App*** *here.*

OMAPL1=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article* ***WL18xx TI Bluetooth Stack AUDM Demo App*** *here.*

MAVRK=*For technical support on MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article* ***WL18xx TI Bluetooth Stack AUDM***

For technical s please post you questions at http://e2e.ti.cor Please post on comments abo article ***WL18xx Bluetooth Sta AUDM Demo A*** *here.*

}}

Bluetooth Stack AUDM Demo App here. ***Demo App*** *here.* ***Demo App*** *here.*

Links

Amplifiers & Linear
Audio
Broadband RF/IF & Digital Radio
Clocks & Timers
Data Converters
DLP & MEMS
High-Reliability
Interface
Logic
Power Management
Processors

- ARM Processors
- Digital Signal Processors (DSP)
- Microcontrollers (MCU)
- OMAP Applications Processors

Switches & Multiplexers
Temperature Sensors & Control ICs
Wireless Connectivity