GENAVBTSNUG

GenAVB/TSN Stack Evaluation User Guide

Rev. 1 — 16 December 2022 **User guide**

Document information

Information	Content
Keywords	GENAVBTSNUG, GenAVB/TSN Stack, Audio Video Bridging (AVB), Time Sensitive Network (TSN), i.MX 6ULL EVK board, i.MX 8MP EVK board, i.MX 93 EVK, i.MX Audio Amplifier, TSN endpoint, AVB Endpoint
Abstract	This document provides information on how to set up Audio Video Bridging evaluation experiments of the GenAVB/TSN Stack on supported NXP hardware platforms

1 Introduction

The GenAVB/TSN Stack is a set of software components that provide Audio Video Bridging (AVB) and Time Sensitive Network (TSN) functionality on NXP SoC and hardware platforms.

This document provides information on how to set up Audio Video Bridging evaluation experiments of the GenAVB/TSN Stack. In that context, it provides information on the SoC and boards that can be used, how to set up the hardware platforms, and information to configure the evaluation software.

2 Initial preparation

2.1 Evaluation boards description and supported roles

The GenAVB/TSN stack is supported on multiple SoCs (i.MX 6 and i.MX 8) and evaluation boards that differ in capabilities. For different use cases, the required connections differ, depending on the ports available on the evaluation boards. This section provides an overall description of the different evaluation boards and their available hardware ports and connections.

The following sections in this document refer to the i.MX evaluation boards depending on their roles:

- **i.MX Audio Amplifier:** A board that can render (and/or decode) audio samples to a speaker connected to a jack output port or an RCA Output port.
- **i.MX Audio Sampler:** A board that can capture audio samples from an analog device connected to an input jack port, MIC, or an RCA Input port
- **i.MX Video Renderer:** A board that can (optionally) demux an MPEG2-TS stream and decode then render video frames to a display.
- **i.MX Audio Video Player:** A board that can demux a MPEG2-TS stream and decode both audio and video frames and render them to their correspondent outputs
- **i.MX Audio Media Server:** A board that can read raw audio samples from a local media file and send them over the network.
- **i.MX Full Media Server:** A board that can read an encoded media file (for example, MP4/MPEG2-TS), demux audio and video, decode audio frames, send separate audio and video AVTP streams, and play the video frames in a local display simultaneously.

Each supported evaluation board can be used in a set of these roles. Refer to the respective board's User Manual to make sure that your board can be used for the desired use case. Also refer to *AN13678* (*i.MX6ULL EVK GenAVB/TSN Rework Application Note* available on Real Time Edge Documentation) for additional information.

2.1.1 i.MX 6ULL EVK board

This evaluation board can support the following roles in the AVB evaluation uses cases described in this document:

- **i.MX Audio Amplifier:** using the Audio Jack as output.
- **i.MX Audio Sampler:** using the MIC as input.
- **i.MX Audio Media Server**

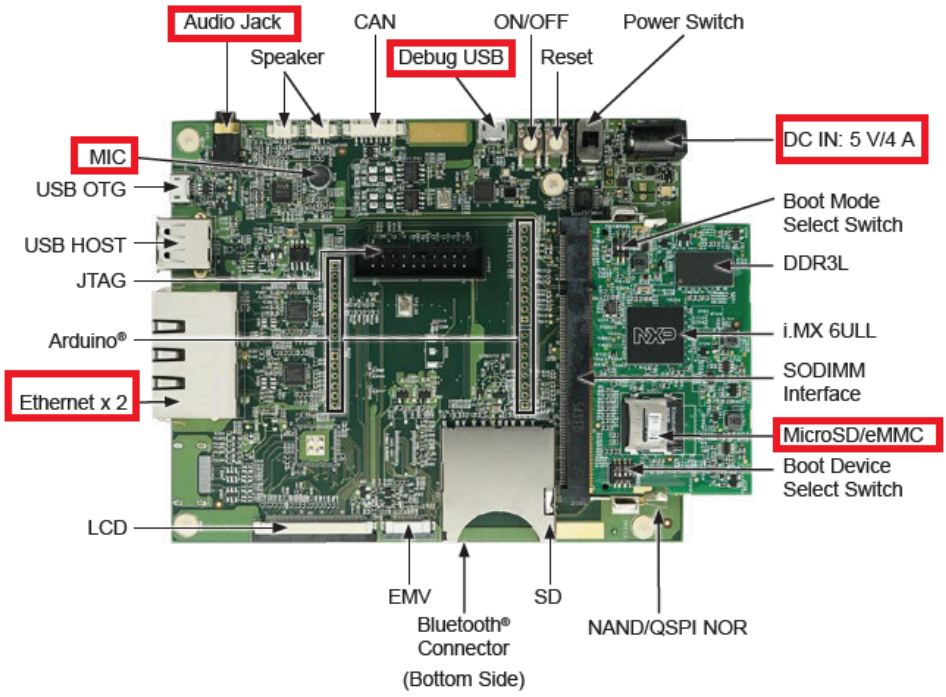

Figure 1. i.MX 6ULL EVK board

2.1.2 i.MX 8M Mini EVK Board

This evaluation board can support the following roles in the AVB evaluation uses cases described in this document:

- **i.MX Audio Amplifier**: using the Audio Jack as output, **with no media clock recovery**
- **i.MX Full Media Server:** using the HDMI port for local display**.**
- **i.MX Audio Media Server.**
- **i.MX Video Renderer:** using the HDMI port as output.
- **i.MX Audio Video Player:** using the Audio Jack as audio output, **with no media clock recovery,** and HDMI for video output.

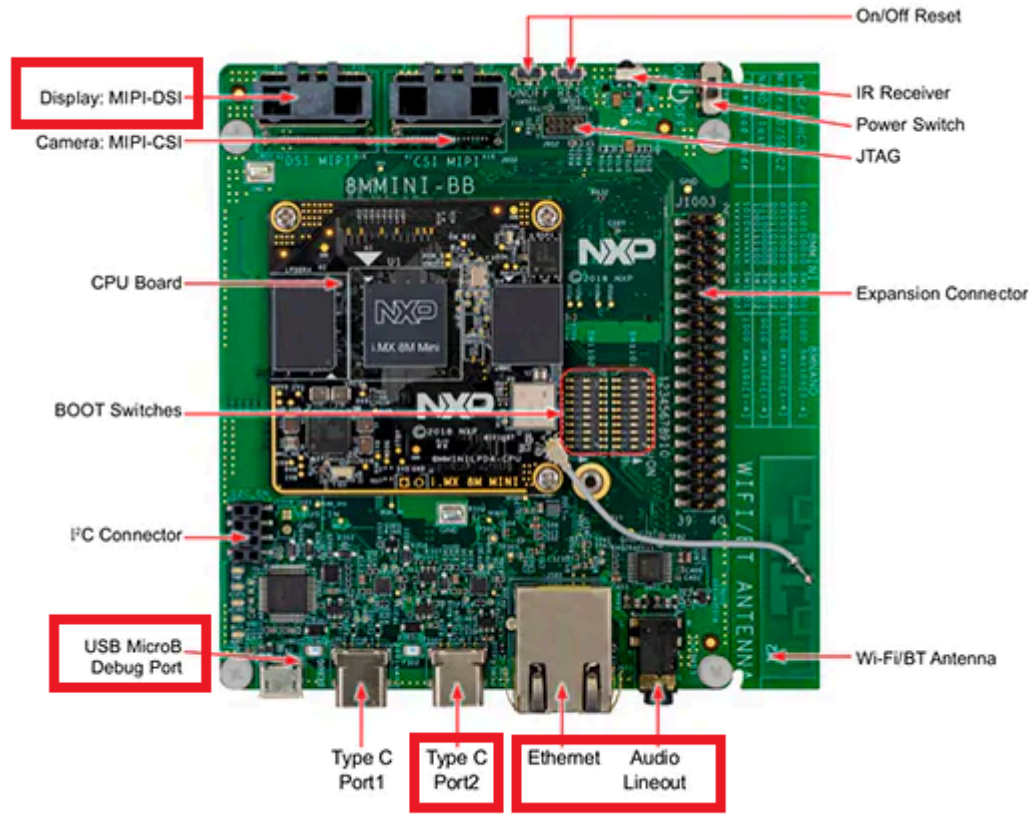

Figure 2. i.MX8M Mini EVK board

2.1.3 i.MX 8M Plus EVK Board

This evaluation board can support the following roles in the AVB evaluation uses cases described in this document:

- **i.MX Audio Amplifier**: using the Audio Jack as output.
- **i.MX Audio Sampler**: using an audio jack input with microphone feature (TRRS with four contacts).
- **i.MX Full Media Server**: using the HDMI port for local display**.**
- **i.MX Audio Media Server**
- **i.MX Video Renderer**: using the HDMI port as output.
- **i.MX Audio Video Player**: using the Audio Jack as audio output, and HDMI for video output.

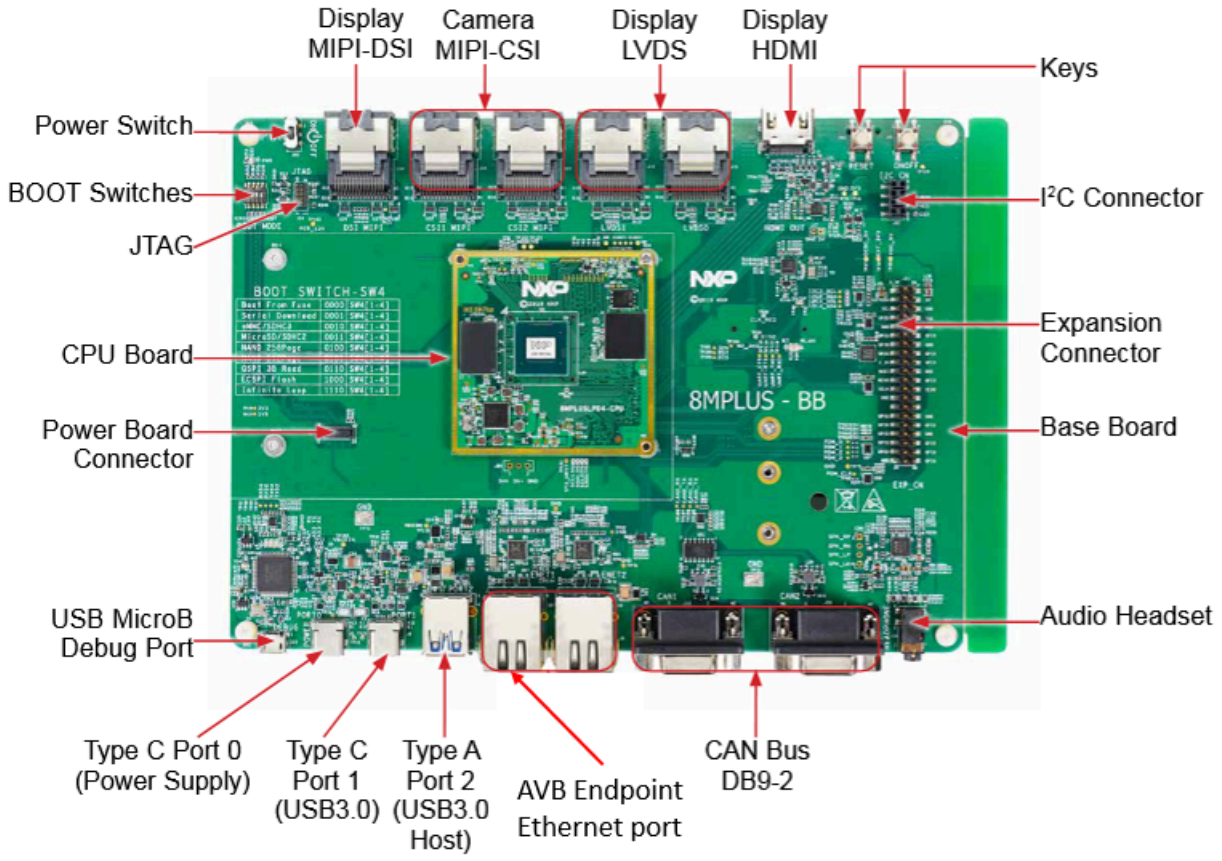

Figure 3. i.MX 8M Plus EVK board

2.1.4 i.MX 93 EVK Board

This evaluation board can support the following roles in the AVB evaluation uses cases described in this document:

• i.MX Audio Amplifier: using the Audio Jack as output, with no media clock recovery

• i.MX Audio Sampler: using an audio jack input with microphone feature (TRRS with four contacts).

• i.MX Audio Media Server.

- **i.MX Audio Amplifier**: using the Audio Jack as output, with no media clock recovery
- **i.MX Audio Sampler:** using an audio jack input with microphone feature (TRRS with four contacts).
- **i.MX Audio Media Server.**

2.2 AVB configuration on evaluation boards

The AVB evaluation package can be configured on various reference boards. Certain specific settings should be performed for ensuring correct AVB operations.

The evaluation boards can support the following roles in the AVB evaluation use cases, which are described in this document:

- i.MX Audio Amplifier: using the Audio Jack as output, with no media clock recovery
- i.MX Audio Sampler: using an audio jack input with microphone feature (TRRS with four contacts).
- i.MX Audio Media Server.

2.2.1 i.MX 6ULL EVK board

This section describes AVB configuration on i.MX 6ULL EVK boards with and without media clock recovery.

2.2.1.1 i.MX 6ULL EVK board with media clock recovery

The i.MX 6ULL EVK board featuring the i.MX 6ULL processor (MCIMX6ULL-EVK) can be used in the evaluation setup as an audio endpoint. When used for a listener role, the board should be slightly modified in order to support the media clock recovery process:

- Connect SD1_DATA2 and GPIO1_IO05 pads (can be done by connecting R1728 and TP2120)
- Connect JTAG_MOD and JTAG_TMS pads (can be done by connecting R1023 and JTAG PIN 7)

For details about the required hardware rework, refer to the *i.MX6ULL EVK GenAVB/TSN Rework Application Note* (*AN13678*) available on Real Time Edge Documentation.

Then, make sure to update the boot parameters to use the device tree binary that includes the hardware description relative to the i.MX 6ULL EVK board, by following the steps listed below.

- Power on the board and stop the automatic boot process by pressing the space bar on the keyboard.
- Enter the following commands at the U-Boot prompt:

```
U-Boot > setenv fdt_file imx6ull-14x14-evk-avb-mcr.dtb
U-Boot > saveenv
U-Boot > boot
```

The change is saved across reboots.

2.2.1.2 i.MX 6ULL EVK board without Media Clock Recovery

For a proper evaluation setup, it is highly recommended to implement the needed AVB hardware rework for the i.MX 6ULL EVK board.

However, a board without the reworks can still run the supported evaluation setups with limited capabilities. In such a case, when used for an audio listener role, the media clock recovery will not be available and no media clock recovery would be launched.

Ensure that the boot parameter is updated to use the device tree binary that includes the hardware description relative to the i.MX 6ULL EVK board:

- Power on the board and stop the automatic boot process by pressing the space bar on the keyboard.

- Enter the following commands at the U-Boot prompt:

```
U-Boot > setenv fdt_file imx6ull-14x14-evk-avb.dtb
U-Boot > saveenv
U-Boot > boot
```

The change is saved across reboots.

2.2.2 i.MX 8MM EVK board

The i.MX 8MM EVK board featuring the i.MX 8M processor (MCIMX8MM-EVK) can be used in any role of the AVB evaluation setup with the limitation that no media clock recovery is supported. Due to a pin conflict, this mechanism cannot be implemented on this evaluation board.

Thus, when used for an audio listener role, the media clock recovery will not be available and no media clock recovery is launched.

Make sure to update the boot parameter to use the device tree binary that includes the hardware description relative to the i.MX 8MM EVK board:

- Power on the board and stop the automatic boot process by pressing the space bar on the keyboard.
- Enter the following commands at the U-Boot prompt:

```
U-Boot > setenv fdtfile imx8mm-evk-avb.dtb
U-Boot > saveenv
U-Boot > boot
```

Note: *depending on the evaluation board revision: REV B or REV C, select the right device tree in U-Boot:*

- `imx8mm-evk-revb-avb.dtb` : for i.MX8MM EVK REVB
- `imx8mm-evk-avb.dtb`: for i.MX8MM EVK REVC

The change is saved across reboots.

2.2.3 i.MX 8MP EVK board

The i.MX 8M Plus EVK board featuring the i.MX 8M Plus application processor can be used in any role of the AVB evaluation setup.

Make sure to update the boot parameter to use the device tree binary that includes the hardware description relative to the i.MX 8M Plus EVK board:

- Power on the board and stop the automatic boot process by pressing the space bar on the keyboard.
- Enter the following commands at the U-Boot prompt:

```
U-Boot > setenv fdtfile imx8mp-evk-avb.dtb
U-Boot > saveenv
U-Boot > boot
```

The change is saved across reboots.

2.2.4 i.MX 93 EVK board

The i.MX 93 EVK board featuring the i.MX 93 application processor can be used as an audio endpoint.

Make sure to update the boot parameter to use the device tree binary that includes the

hardware description relative to the i.MX 93 EVK board:

- Power on the board and stop the automatic boot process by pressing the space bar on the keyboard.
- Enter the following commands at the U-Boot prompt:

```
U-Boot > setenv fdtfile imx93-11x11-evk-avb.dtb
U-Boot > saveenv
U-Boot > boot
```

The change is saved across reboots

2.3 Configuring GenAVB/TSN stack and demo applications

The i.MX 93 EVK board featuring the i.MX 93 application processor can be used as an audio endpoint. Make sure to update the boot parameter to use the device tree binary that includes the hardware description relative to the i.MX 93 EVK board:

For some hardware platforms, the GenAVB/TSN stack supports both modes: endpoint TSN and Endpoint AVB. To configure the stack to Endpoint AVB mode, the file `/etc/genavb/config_avb` should set the `GENAVB_TSN_CONFIG` parameter to the right configuration. This can be done using the following steps:

```
# avb.sh stop_all
# vi /etc/genavb/config
```

For platforms that support only Endpoint AVB (for example, i.MX 8MM and i.MX 6ULL) use the value:

```
GENAVB_TSN_CONFIG=1
```

For platforms supporting both Endpoint AVB and Endpoint TSN (for example, i.MX 8MP and i.MX 93) use the value:

```
GENAVB_TSN_CONFIG=2
```

Additionally, the stack is not started automatically on boot. But, a systemd service can be enabled to assure automatic stack startup (on next reboot) with the following commands:

1. Power on the board and stop the automatic boot process by pressing the space bar on the keyboard.
2. Enter the following commands at the U-Boot prompt:

```
U-Boot > setenv fdtfile imx93-11x11-evk-avb.dtb
U-Boot > saveenv
U-Boot > boot
```

The change is saved across reboots.

2.3.1 Profiles supported by GenAVB/TSN stack

For some hardware platforms, the GenAVB/TSN stack supports both modes: endpoint TSN and Endpoint AVB.

To configure the stack to Endpoint AVB mode, use the file `/etc/genavb/config` to set the `GENAVB_TSN_CONFIG` parameter to the right configuration:

```
# avb.sh stop_all
# vi /etc/genavb/config
```

For platforms supporting only Endpoint AVB (such as i.MX 8M Mini and i.MX 6ULL), use the value below:

```
GENAVB_TSN_CONFIG=1
```

For platforms (such as i.MX 8M Plus and i.MX 93) supporting both Endpoint AVB and Endpoint TSN, use the value below:

```
GENAVB_TSN_CONFIG=2
```

2.3.2 Configuring GenAVB/TSN stack to start at system boot

By default, the stack does not start automatically on boot. But, a systemd service can be enabled to assure automatic stack start up (on next reboot) using the following commands:

```
# systemctl enable genavb-tsn
# systemctl daemon-reload
```

2.3.3 Profiles supported by GenAVB/TSN stack

The following sections describe the various profiles supported by the GenAVB/TSN stack as AVB endpoint. The change from one profile to another is made by modifying the `/etc/genavb/config_avb` file.

This file specifies a pair of configuration files:

- `APPS_CFG_FILE` (`apps-*.cfg`) points to a file containing a demo configuration (media application to use, controller option, or so on). It is parsed by the startup script `avb.sh`.
- `GENAVB_CFG_FILE` (`genavb-*.cfg`) points to a file containing the configuration of the AVB stack and is parsed by the AVB application.

A demo profile comprises a pair of `cfg` files.

The file `/etc/genavb/config_avb` already groups the `cfg` files by pairs, the two lines corresponding to the desired demo profile should be uncommented.

3 AVB Audio Sampler/Amplifier Back-to-Back

This section describes the i.MX platform supporting AVB Audio Talker and Listener roles on two i.MX evaluation boards connected back-to-back.

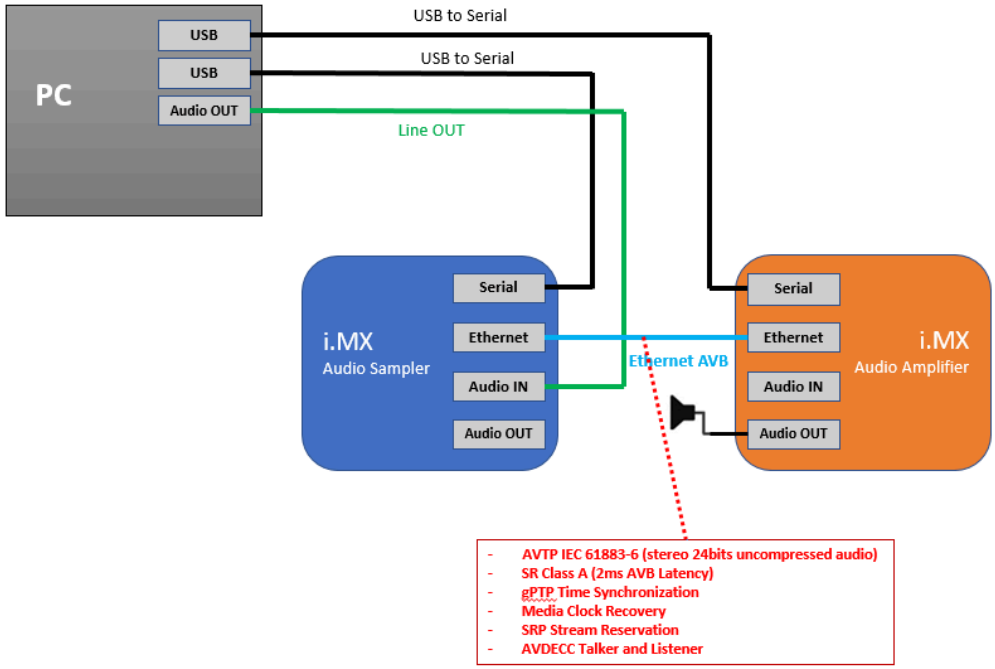

Figure 4. AVB i.MX Back-to-Back setup

3.1 Requirements

1. One **i.MX Audio Amplifier** capable evaluation board.
2. One **i.MX Audio Sampler** capable evaluation board.
3. Headphones/speakers with male Jack.

Figure 5. Headphones used as speakers

4. Two USB/Serial cables.
5. Windows OS laptop.

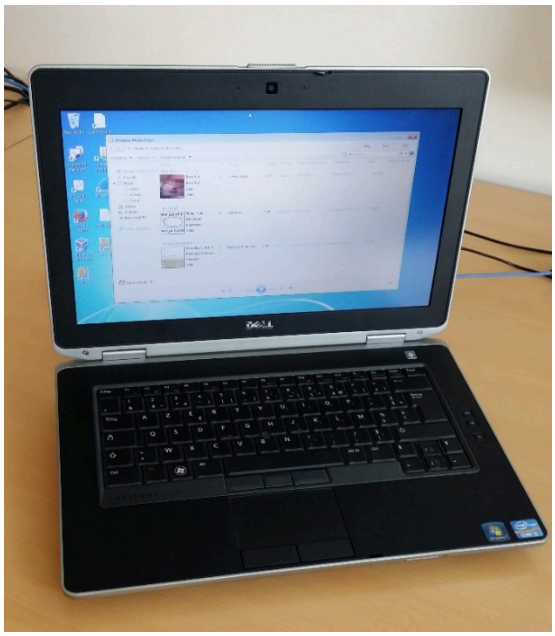

Figure 6. Laptop running Windows OS

3.2 Setup preparation

1. Connect the headphones/speakers to the **i.MX Audio Amplifier** audio Jack port.
2. Connect the Line OUT of the laptop to the available audio input (audio jack port with microphone feature or on board MIC) of the i.MX Audio Sampler.
3. Connect both the i.MX boards with an Ethernet RJ45 cable.
4. Connect a Serial/USB cable to each i.MX board and to some USB ports of the Laptop.
5. Install a Terminal emulator on the Laptop for enabling console display of the i.MX boards through the serial/USB ports, as shown in the figure below.

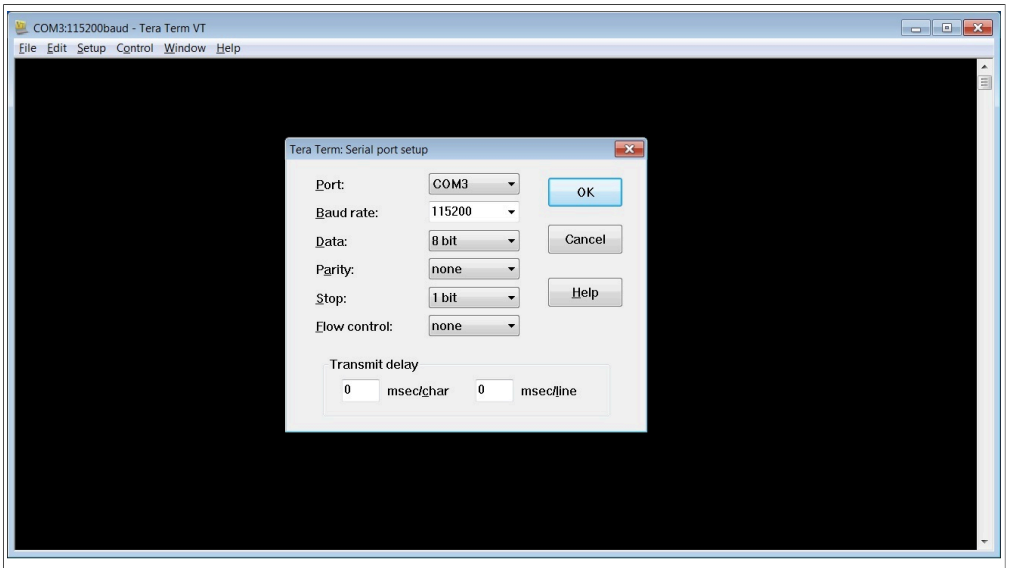

Figure 7. Connecting to the i.MX console via a terminal

The i.MX boards should now be configured for supporting media clock functions. The board acting as talker (sampler) should be configured to support media clock generation. The board acting as listener (amplifier) should be configured to support media clock recovery. Refer to the section Initial Preparation for completing this preparation, depending on the board type used for evaluation.

3.2.1 Preparing the AVB configuration for the Talker

The default AVB script must be modified to configure operations of the Talker entity as using a custom Media Application. The AVB Stack is provided with an ALSA application, supporting audio sampling from an ALSA interface. To enable this media application, the GenAVB/TSN avb configuration file needs to be modified as follows:

1. Power on the i.MX board and let the boot process complete.
2. Edit the GenAVB/TSN avb configuration file using the following command at the Linux prompt:

```
# vi /etc/genavb/config_avb
```

3. Set the configuration profile to PROFILE 14:

```
PROFILE=14
```

4. Exit and save the file then reboot the board. The change is saved across reboots, so it is required to be performed only once.

The setup is then ready for evaluation.

3.2.2 Preparing the AVB configuration for the listener

The default AVB script must be modified to configure operations of the Listener entity as using a custom Media Application. The AVB Stack is provided with an ALSA application example, interfaced to the AVB stack through the GenAVB/TSN API, and supporting audio playout to an ALSA interface. To enable this media application, the GenAVB/TSN avb configuration file needs to be modified as follows:

1. Power on the i.MX board and let the boot process complete.
2. Edit the GenAVB/TSN avb configuration file using the following command at the Linux prompt:

```
 # vi /etc/genavb/config_avb
```

3. Set the configuration profile to PROFILE 15:

```
PROFILE=15
```

4. Exit and save the file and then reboot the board.
5. The change is saved across reboots, so this is required to be done only once.

3.3 Evaluation instructions

On both talker and listener boards, power on and boot the boards, and then login as root (no password). The AVB stack is automatically started (per `genavb-tsn` systemd service configuration described in Section 2.3.2).

The AVB audio stream connection is established automatically. When PTP is synchronized between the two systems, a media player on the laptop can be used to play audio to its line out interface. Audio is rendered on the headset of the listener board.

4 AVB Audio Media Server/Amplifier Back-to-back

This section describes the i.MX platform supporting AVB Audio Media Server and Amplifier roles on two i.MX boards connected back-to-back. This setup uses custom media applications interfaced to the AVB Stack through the GenAVB/TSN API.

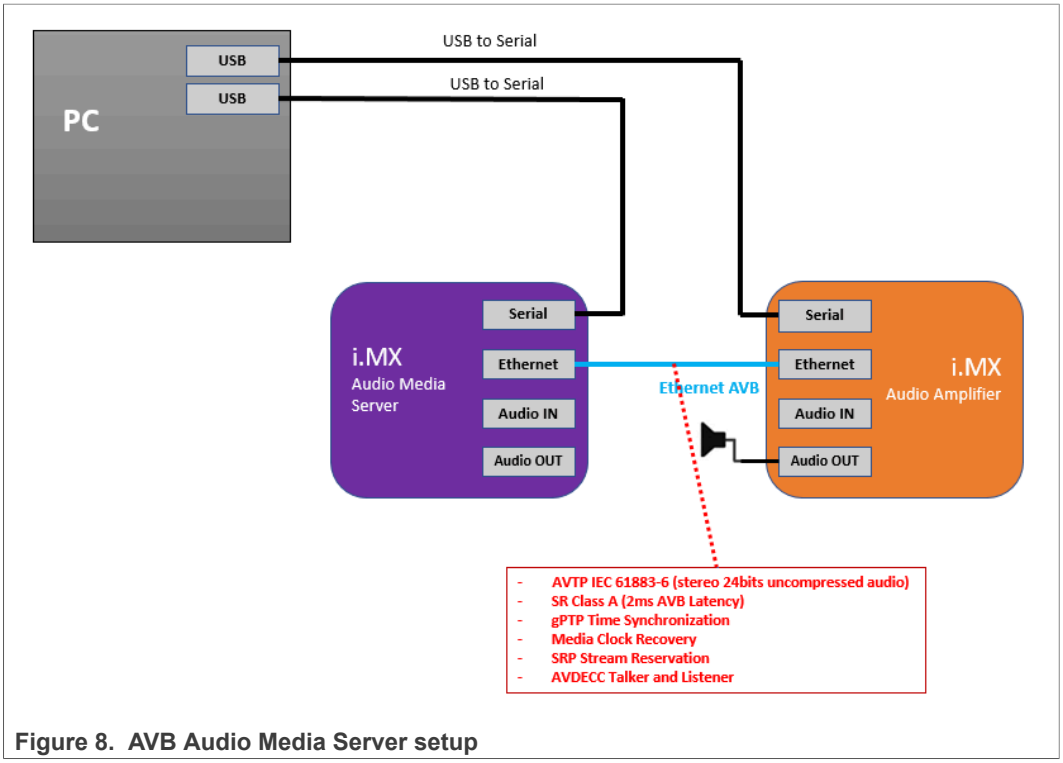

Figure 8. AVB Audio Media Server setup

4.1 Requirements

The section requirement is similar to the AVB Audio Sampler/Amplifier Back-to-Back setup, with the following variations:

- The i.MX talker is a Media Server (**i.MX Audio Media Server** capable board) reading samples from a file stored on the SD flash memory.
- The talker does not need to support media clock functions.

4.2 Setup preparation

The setup preparation is similar to the AVB Audio Sampler/Amplifier Back-to-Back setup. The variation is that there is no need to connect an audio analog device to the audio input of the Talker board, as the samples to be played out are stored in a file of the SD flash memory. For this reason, configuring the talker board for supporting media clock generation is not required for this use case.

4.2.1 Preparing AVB configuration for the Talker

The default AVB script should be modified to configure operations of the Talker entity as using a custom Media Application. The AVB Stack is provided with a simple Media Server application example, interfaced to the AVB stack through the GenAVB/TSN API,

and supporting reading audio samples from a media file. To enable using this media application, the GenAVB/TSN avb configuration file needs to be modified as follows:

1. Power ON the i.MX board and let the boot process complete.
2. Edit the GenAVB/TSN avb configuration file using the following command at the Linux prompt:

```
# vi /etc/genavb/config_avb
```

3. Set the configuration profile to PROFILE 9:

```
PROFILE=9
```

4. Exit and save the file then reboot the board.

The change is saved across reboots, so this has only to be done once. The setup is then ready for evaluation.

4.2.2 Preparing the AVB configuration for the Listener

The default AVB script needs to be modified to configure operations of the Listener entity as using a custom Media Application. The AVB Stack is provided with an ALSA application example, interfaced to the AVB stack through the GenAVB/TSN API, and supporting audio playout to the ALSA interface. To enable using this media application, the GenAVB/TSN avb configuration file needs to be modified as follows:

1. Power on the i.MX board and let the boot process complete.
2. Edit the GenAVB/TSN avb configuration file using the following command in the Linux prompt:

```
# vi /etc/genavb/config_avb
```

3. Set the configuration profile to PROFILE 11:

```
PROFILE=11
```

4. Exit and save the file then reboot the board.

The change is saved across reboots, so this has only to be done once.

4.3 Evaluation instructions

On both Talker and Listener boards, power on and boot the boards, and log in to the boards as root (no password). The AVB stack automatically starts (as per `genavb-tsn` systemd service configuration described in Section 2.3.2).

The AVB audio stream connection is automatically established between Talker and Listener. At that point, the Media Server application of the Talker starts streaming the audio samples from the source audio file to the AVB Stack. The audio file is read indefinitely in a loop. Samples are transported as an AVB stream to the Listener and delivered to the ALSA application for playout to the audio device. Audio is rendered on the headset or speakers connected to the listener board.

The evaluation can be stopped by using the following command on either the Talker or the Listener side:

```
# systemctl stop genavb-tsn
```

It can be re-started using the following command on either the Talker or the Listener side:

```
# systemctl start genavb-tsn
```

4.4 Audio file format and generation

The current Media Server application uses a raw audio format:

- Two channels
- 48 kHz
- 24 bits
- Big endian numbering format

An **i.MX Audio Sampler** capable board can be used to generate such a file, by connecting an analog source (external player) to a Jack audio input port.

The following ALSA command allows capturing and storing the file in the expected format:

```
# arecord_-D <device> –t raw –c 2 –r 48000 –f S24_BE <file
name>
```

To verify and listen to the captured file, connect a hearing device (headphones/speakers) to audio output Jack port or MIC. (In case of the SABRE-AI board, use the RCA/Jack cable connected to Audio Line OUT port.)

Use the following ALSA command:

```
# aplay -D <device> –c 2 –r 48000 –f S24_BE <file name>
```

The alsa device should be assigned to audio codec connected to jack interface (wm8960-audio).

5 AVB Audio Multi-Stream

This section describes i.MX platform supporting AVB Audio Media Server and Listener roles on three i.MX boards connected through an AVB switch.

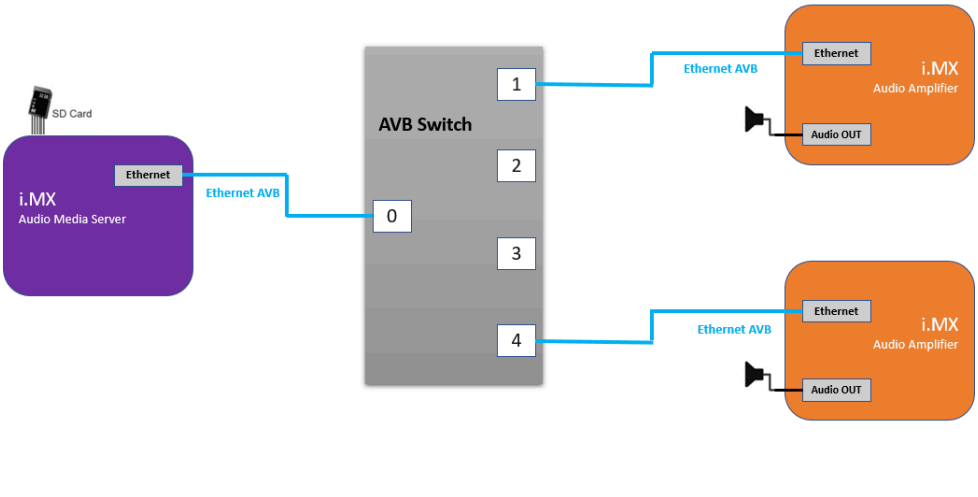

Figure 9. AVB Audio Multi-Stream setup

5.1 Requirements

The setup requirement is as follows:

- The i.MX talker is an Audio Media Server reading samples from some files stored on the SD flash memory. The platform is similar to the talker used in the AVB Audio Media Server setup.
- The i.MX listeners are Audio Amplifiers similar to the listener used in the AVB Audio Sampler/Amplifier Back-to-Back setup. Up to four listeners can be part of the evaluation setup.
- An AVB switch is required to interconnect the talker and listeners nodes.

5.2 Setup preparation

The hardware preparation of the i.MX platforms is similar to the AVB Audio Media Server setup, with the variation that several listeners may be prepared. All those endpoints should be connected to an AVB switch. The configuration of the AVB stack on talker and listeners is specific to this evaluation and should be completed as described in the next sections.

5.2.1 Preparing the AVB configuration for the Talker

The default AVB script needs to be modified to configure operations of the Talker entity as using a custom Media Application. The AVB Stack is provided with a multi-stream Media Server application example, interfaced to the AVB stack through the GenAVB/TSN API, and supports reading audio samples from media files. The multi-stream application uses an AEM profile supporting streaming of up to 8 audio streams.

To enable this media application, the GenAVB/TSN avb configuration file should be modified as follows:

1. Power on the i.MX board and let the boot process complete.

2. Edit the GenAVB/TSN avb configuration file using the following command at the Linux prompt:

```
# vi /etc/genavb/config_avb
```

3. Set the configuration profile to PROFILE 7:

```
PROFILE=7
```

4. Exit and save the file then reboot the board.
5. A raw audio file `sample1.raw` is available in the `/home/media` repository. The multi-stream application example looks for audio files named `talker_mediaX.raw` in the `/home/media` repository, with `X` being the stream number (i.e. stream #0 will read `media0.raw` file…). Hence, before executing the multi-stream application, some symbolic links needs to be created in the `/home/media` directory for associating the `talker_mediaX.raw` names to the `sample1.raw` file, or any other raw audio file that would be present in the file system:

```
# ln –s sample1.raw talker_media0.raw
# ln –s sample1.raw talker_media1.raw
# ln –s sample1.raw talker_media2.raw
# ln –s sample1.raw talker_media3.raw
```

6. Check the audio files links:

```
# ls -l
total 10708
lrwxrwxrwx  1 root root 11 Jan  1 00:18 talker_media0.raw ->
 sample1.raw
lrwxrwxrwx  1 root root 11 Jan  1 00:19 talker_media1.raw ->
 sample1.raw
lrwxrwxrwx  1 root root 11 Jan  1 00:20 talker_media2.raw ->
 sample1.raw
lrwxrwxrwx  1 root root 11 Jan  1 00:20 talker_media3.raw ->
 sample1.raw
-rwxr--r--  1 root root 6000000 Feb  6  2015 sample1.raw
```

Reboot the board. The change is saved across reboots, so this needs to be performed only once.

7. The setup is now ready for evaluation.

5.2.2 Preparing the AVB configuration for the Listeners

The default AVB script needs to be modified to configure operations of the listener entity to use the ALSA application, supporting audio playout to the ALSA interface. To enable this media application, the GenAVB/TSN configuration file should be modified as per the steps described below:

1. Power on the i.MX board and let the boot process complete.
2. Edit the `GenAVB/TSN avb` configuration file using the following command at the Linux prompt:

```
# vi /etc/genavb/config_avb
```

3. Set the configuration profile to `PROFILE 11`:

```
PROFILE=11
```

4. Then, edit the `/etc/genavb/genavb-listener-btb.cfg` configuration file to set the talker's AVDECC Entity ID and Unique ID information. This step is mandatory to

configure the stream output of the talker, the current listener being configured would connect to.
In this use case, each listener connects to a different AVB stream. Therefore, each listener in the setup should connect to a different talker's Unique ID, with audio transported in separate AVB streams. For this, uncomment and edit the following fields in the AVDECC entity 1 section with X being the entity , and Y being the unique ID:

```
[AVB_AVDECC_ENTITY_1]
…
talker_entity_id_list = X
…
talker_unique_id_list = Y
```

Note: *1) The Talker entity ID is the unique EUI-64 identifier of the Talker AVDECC entity. It is derived from the EUI-48 MAC address of the Talker padded with the entity index value. For instance, assuming a Talker has a MAC address set as 00:11:22:33:44:55 and its entity index within the AVDECC configuration is 0x0000, then its corresponding entity ID would be:* `talker_entity_id_list = 0x0011223344550000`
The Talker Unique ID identifies the stream source in the talker's entity model, usually an index starting from 0. For instance, to connect to the stream output 0 the corresponding talker unique ID would be:

```
talker_unique_id_list = 0
```

Note: *2)The talker entity information can be displayed by using the AVDECC controller application available on the talker endpoint:*

```
root@imx6qsabreauto-avb:~# genavb-controller-app -l
NXP's GenAVB AVDECC controller demo application
Number of discovered entities: 2
Entity ID = 0x49f04433f0001     Model ID = 0x49f04433f0001
     MAC address: 00:04:9F:04:43:3F     Capabilities = 0x8
 Association ID = 0x2
Controller

Controls:
         None
Entity ID = 0x49f04433f0000     Model ID = 0x49f04433f0001
    MAC address: 00:04:9F:04:43:3F     Capabilities = 0x708
 Association ID = 0x2
     Talker:     sources = 8     capabilities = 0x4801

Stream 0: name = Stream output 0 number of formats = 1 flags
 = 0x6 current_format = 0x00a0020240000200 ( 61883-6 AM824
 2chans 48000kHz )
Stream 1: name = Stream output 1 number of formats = 1 flags
 = 0x6 current_format = 0x00a0020240000200 ( 61883-6 AM824
 2chans 48000kHz )
Stream 2: name = Stream output 2 number of formats = 1 flags
 = 0x6 current_format = 0x00a0020240000200 ( 61883-6 AM824
 2chans 48000kHz )
Stream 3: name = Stream output 3 number of formats = 1 flags
 = 0x6 current_format = 0x00a0020240000200 ( 61883-6 AM824
 2chans 48000kHz )
Stream 4: name = Stream output 4 number of formats = 1 flags
 = 0x6 current_format = 0x00a0020240000200 ( 61883-6 AM824
 2chans 48000kHz )
```

```
Stream 5: name = Stream output 5 number of formats = 1 flags
 = 0x6 current_format = 0x00a0020240000200 ( 61883-6 AM824
 2chans 48000kHz )
Stream 6: name = Stream output 6 number of formats = 1 flags
 = 0x6 current_format = 0x00a0020240000200 ( 61883-6 AM824
 2chans 48000kHz )
Stream 7: name = Stream output 7 number of formats = 1 flags
 = 0x6 current_format = 0x00a0020240000200 ( 61883-6 AM824
 2chans 48000kHz )
Controls:
         None
```

This command should be entered while AVB is running on the endpoints part of the AVB setup.

5. Exit and save the file. Then, reboot the board. Repeat on all the listeners. The changes are saved across reboots, so this needs to be performed only once.

5.3 Evaluation instructions

On both talker and listener boards, power on and boot the boards, and log in to the boards as root (no password). The AVB stack is automatically started (as per `genavb-tsn` systemd service configuration described in Section 2.3.2).

The listeners should connect automatically to the different streams advertised by the talker.

On the talker side, the media application log can be monitored using following command:

```
# tail -f /var/log/avb_media_app
```

The log indicates that several streams are configured in parallel with different stream IDs:

```
AVB_MSG_MEDIA_STACK_CONNECT
stream ID: 00049f0254af0003
find_free_thread: thread(0x13bf8) found
find_free_stream: stream(0x13c58) found
mclock_gen_ptp_config: wake period : 72/48000 s = 1500000 ns
Configured AVB batch size (bytes): 1008
stream ID: 00049f0254af0003
media file name: /home/media/talker_media3.raw
mode: TALKER
msg_send(0x13bf8, 1)
msg_receive(0x13bf8, 1)
thread_add_stream: thread(0x13bf8) added stream(0x13c58) fd(7)
AVB_MSG_MEDIA_STACK_CONNECT
stream ID: 00049f0254af0001
find_free_thread: thread(0x13d38) found
find_free_stream: stream(0x13d98) found
Configured AVB batch size (bytes): 1008
stream ID: 00049f0254af0001
media file name: /home/media/talker_media1.raw
mode: TALKER
msg_send(0x13d38, 1)
msg_receive(0x13d38, 1)
thread_add_stream: thread(0x13d38) added stream(0x13d98) fd(10)
```

Each listener playouts music on the regular audio output.

The evaluation can be stopped using the following command on either talker or listeners side:

```
# systemctl stop genavb-tsn
```

It can be re-started using the following command on either talker or listeners side:

```
# systemctl start genavb-tsn
```

5.4 Audio file format and generation

The audio file format and generation is similar to the AVB Audio Media Server setup.

6 AVB Audio Multi-Format

This section describes i.MX platform supporting AVB Audio Multi-format on several i.MX boards connected in back-to-back or through an AVB switch.

6.1 Requirements

The setup requirement is as follows:

- The i.MX talker is an Audio Sampler reading samples from a live input source. The platform is similar to the talker used in the AVB Audio Sampler/Amplifier Back-to-Back setup.
- The i.MX listeners are Audio Amplifiers similar to the listener used in the AVB Audio Sampler/Amplifier Back-to-Back setup. Up to four listeners can be part of the evaluation setup.
- An AVB switch is required to interconnect the talker and listeners nodes, unless there is only one listener.

Note that the i.MX 6ULL EVK board does not support 96 kHz (and above) formats.

6.2 Setup preparation

The hardware preparation of the i.MX platforms is similar to the AVB Audio Sampler/Amplifier Back-to-Back setup, with evolution that several listeners may be prepared. All those endpoints should be connected to an AVB switch. The configuration of the AVB stack on talker and listeners is specific to this evaluation and should be completed as described in the next sections.

6.2.1 Preparing the AVB configuration for the Talker

The default AVB script needs to be modified to configure operations of the Talker entity as using a custom Media Application. The AVB Stack is provided with a multi-stream Media Server application example, interfaced to the AVB stack through the GenAVB/TSN API, and supporting reading audio samples from media files. The GenAVB/TSN media application uses an AEM profile supporting streaming of 8 different audio streams. Please note that only one of those streams can run at a time as they all have different, non-compatible settings.

To enable this media application, the `GenAVB/TSN avb` configuration file needs to be modified as follows:

1. Power on the i.MX board and let the boot process complete.
2. Edit the `GenAVB/TSN avb` configuration file using the following command at the Linux prompt:

```
# vi /etc/genavb/config_avb
```

3. Set the configuration profile to PROFILE 19:

```
PROFILE=19
```

4. Then it is possible to edit the `/etc/genavb/srp.cfg` configuration file to set the talker's enabled SR class. Two classes are needed to run correctly. The configuration file associated to the profile can be modified at any moment, but changes would only be effective after AVB is restarted. SR classes can be enabled by setting "`sr_class_enabled`", which can take any combination of two existing classes, separated by a comma. For example:
   `"A,B"` => enables `SR_CLASS_A` and `SR_CLASS_B`
   `"E,B"` => enables `SR_CLASS_B` and `SR_CLASS_E`
   For any combination, class "HIGH" is the first enabled class in an alphabetical order and class "LOW" the second one. So "A,B" is identical to "B,A".
   It is not allowed to:
   - Enable less/more than 2 classes (for example: "A";"B,C,D")
   - Enable a class twice (for example: "B,B")
   - Enable an unknown class

   If those conditions are not followed, an error message is displayed in the logs and the AVB starting process stops.
   Note: *Backup the original `/etc/genavb/srp.cfg` before changing it so that the original configuration can be restored at the end of the current evaluation.*
5. Edit `/etc/genavb/apps-listener-talker-multi-format.cfg` to setup the AVDECC entity's stream. The configuration should be:

```
CFG_EXTERNAL_MEDIA_APP_OPT='${CFG_MEDIA_CLOCK} -C
 ${CFG_ALSA_CAPTURE_DEVICE} -P ${CFG_ALSA_PLAYBACK_DEVICE} -
T -A 0 -z 0 -T -A 1 -z 0 -T -A 2 -z 0 -T -A 3 -z 0 -T -A 4 -
z 0 -T -A 5 -z 0 -T -A 6 -z 0 -T -A 7 -z 0'
```

   This configuration maps all AVDECC talker streams to the same ALSA handler 0 and therefore, the same ALSA capture device.
6. Exit and save the file. Then reboot the board. The change is saved across reboots so this is required to be done only once.

The setup is then ready for evaluation.

6.2.2 Preparing the AVB configuration for the Listeners

The default AVB script needs to be modified to configure operations of the listener entity as using the ALSA application, supporting audio playout to the ALSA interface. To enable this media application, the GenAVB/TSN avb configuration file needs to be modified as follows:

1. Power on the i.MX board and let the boot process complete.

2. Edit the GenAVB/TSN avb configuration file using the following command at the Linux prompt:

```
# vi /etc/genavb/config_avb
```

3. Set the configuration profile to PROFILE 19:

```
PROFILE=19
```

4. Edit the `/etc/genavb/srp.cfg configuration` file to set the listener's enabled SR class. Refer to the talker configuration in the previous section for details.
5. Edit `/etc/genavb/apps-listener-talker-multi-format.cfg` to setup the AVDECC entity's stream. The configuration should be:

```
CFG_EXTERNAL_MEDIA_APP_OPT="-s -L -A 0 -z 0 -L -A 1 -z 0 -L -
A 2 -z 0 -L -A 3 -z 0 -L -A 4 -z 0 -L -A 5 -z 0 -L -A 6 -z 0
 -L -A 7 -z 0"
```

This configuration maps all AVDECC listener streams to the same ALSA handler 0 and hence, the same ALSA playback device.
Use the below configuration to set the listener as media clock slave:

```
-----------------------------------
CFG_MEDIA_CLOCK="-L -S 0 -c 0"
-----------------------------------
```

6. Exit and save the file. Then, reboot the board.
Note: The talker entity information can be displayed out by using the AVDECC controller application available on the talker endpoint:

```
root@imx6qsabreauto-avb:~# genavb-controller-app -l
NXP's GenAVB AVDECC controller demo application
Number of discovered entities: 2
Entity ID = 0x49f039dc00001     Model ID = 0x49f0000080001
     MAC address: 00:04:9F:00:00:07     Capabilities = 0x8
 Association ID = 0x0
     Controller
     Controls:
         None
Entity ID = 0x49f039dc00000     Model ID = 0x49f0000070001
    MAC address: 00:04:9F:00:00:07     Capabilities = 0x708
 Association ID = 0x0
```

```
Talker: sources = 8 capabilities = 0x4801
Stream 0: name = Stream output 0 number of formats = 1 flags =
0x6 current_format = 0x00a0020240000200 ( 61883-6 AM824 2chans
48000Hz
```

```
)
         Stream   1: name =      Stream output 1
 number of formats = 1    flags = 0x6    current_format =
 0x00a0040240000200 ( 61883-6 AM824 2chans 96000Hz
)
         Stream   2: name =      Stream output 2
 number of formats = 1    flags = 0x6    current_format =
 0x0205021800806000 ( AAF 2chans 24/32bits 48000Hz
6samples/packet )
```

```
         Stream   3: name =      Stream output 3
 number of formats = 1    flags = 0x6    current_format =
 0x020502180080c000 ( AAF 2chans 24/32bits 48000Hz
12samples/packet )
         Stream   4: name =      Stream output 4
 number of formats = 1    flags = 0x6    current_format =
 0x0205021800840000 ( AAF 2chans 24/32bits 48000Hz
64samples/packet )
         Stream   5: name =      Stream output 5
 number of formats = 1    flags = 0x6    current_format =
 0x0205021800830000 ( AAF 2chans 24/32bits 48000Hz
48samples/packet )
         Stream   6: name =      Stream output 6
 number of formats = 1    flags = 0x6    current_format =
 0x020702180080c000 ( AAF 2chans 24/32bits 96000Hz
12samples/packet )
         Stream   7: name =      Stream output 7
 number of formats = 1    flags = 0x6    current_format =
 0x0209021800818000 ( AAF 2chans 24/32bits 192000Hz
 24samples/packet )
     Listener:   sinks   = 8     capabilities = 0x4801
         Stream   0: name =       Stream input 0
 number of formats = 1    flags = 0x6    current_format =
 0x00a0020240000200 ( 61883-6 AM824 2chans 48000Hz
)
         Stream   1: name =       Stream input 1
 number of formats = 1    flags = 0x6    current_format =
 0x00a0040240000200 ( 61883-6 AM824 2chans 96000Hz
)
         Stream   2: name =       Stream input 2
 number of formats = 1    flags = 0x6    current_format =
 0x0205021800806000 ( AAF 2chans 24/32bits 48000Hz
6samples/packet )
         Stream   3: name =       Stream input 3
 number of formats = 1    flags = 0x6    current_format =
 0x020502180080c000 ( AAF 2chans 24/32bits 48000Hz
12samples/packet )
         Stream   4: name =       Stream input 4
 number of formats = 1    flags = 0x6    current_format =
 0x0205021800840000 ( AAF 2chans 24/32bits 48000Hz
64samples/packet )
         Stream   5: name =       Stream input 5
 number of formats = 1    flags = 0x6    current_format =
 0x0205021800830000 ( AAF 2chans 24/32bits 48000Hz
48samples/packet )
         Stream   6: name =       Stream input 6
 number of formats = 1    flags = 0x6    current_format =
 0x020702180080c000 ( AAF 2chans 24/32bits 96000Hz
12samples/packet )
         Stream   7: name =       Stream input 7
 number of formats = 1    flags = 0x6    current_format =
 0x0209021800818000 ( AAF 2chans 24/32bits 192000Hz
 24samples/packet )
     Controls:
         Control  0: name =     Volume Control 0    type =
 0x90e0f00000000004    read-only =  No    value_type = 1  min
 = 0 current = 100 max = 100 step = 1
```

7. This command should be entered while AVB is running on the endpoints part of the AVB setup. Repeat on all the listeners.

The changes are saved across reboots, so the process needs to be performed only once.

6.3 Evaluation instructions

On both talker and listener boards, power on and boot the boards, and log in to the boards as root (no password). The AVB stack is automatically started (as per `genavb-tsn` systemd service configuration described in Section 2.3.2).

To connect streams, use the following command:

```
genavb-controller-app -c <talker_entity_id> <talker_unique_id>
 <listener_entity_id> <listener_unique_id> <flag>
```

`<flag>` allows to choose between the high- and low-class priority.

For example, if *CLASS_A* and *CLASS_C* are enabled, `flag=0` connects a stream using CLASS_A and `flag=1` connects a stream using CLASS_C.

It is important to have **only one stream connected at a time** as all streams are mapped to the same ALSA device (playback or capture) as specified in above configurations.

Also, the talker and listener stream ID should match in order to have the same stream formats connected.

To disconnect a stream, use the command:

```
genavb-controller-app -d < talker_entity_id >
 <talker_unique_id> <listener_entity_id> <listener_unique_id>
```

These AVDECC stream profile definitions are optimized for specific SR classes. Here is a summary of the stream profile classification:

Table 1. AVDECC stream profile definitions and SR classes

Stream number	Rate	Format	Class supported
0	48000 Hz	61883-6 AM824	A,B,C,E
1	96000 Hz	61883-6 AM824	A,B,C,E
2	48000 Hz	AAF	A
3	48000 Hz	AAF	B
4	48000 Hz	AAF	C
5	48000 Hz	AAF	E
6	96000 Hz	AAF	A
7	192000 Hz	AAF	A

The evaluation can be stopped using the following command on either the talker or listener's side:

```
# systemctl stop genavb-tsn
```

It can be restarted using the following command on either the talker or listener's side:

```
```

```
# systemctl start genavb-tsn
```

Note that any change in the enabled SR class configuration, should also be followed by manually restarting the TSN process alongside the AVB process above:

```
# tsn.sh restart
```

6.4 Audio file format and generation

The audio file format and generation is similar to the AVB Audio Media Server setup.

7 AVB Audio/Video Media Server

This section describes AVB Audio/Video Media Server and Players, implemented on i.MX evaluation boards, connected through an AVB switch.

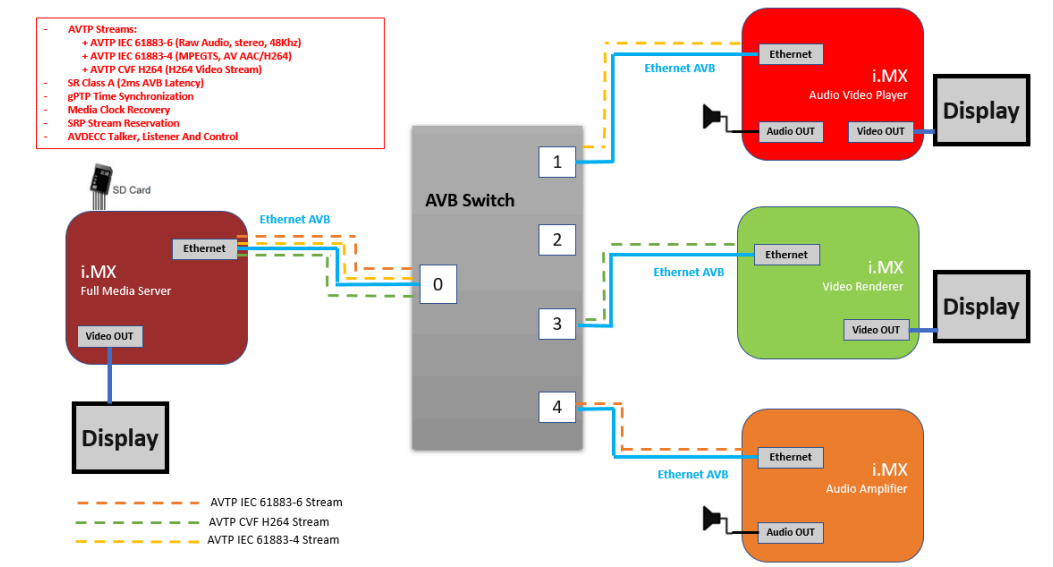

Figure 10. AVB Audio/Video Media Server setup

7.1 Requirements

The setup requirement is as follows:

- The i.MX talker is an **i.MX Full Media Server** capable board reading samples from files stored on the SD flash memory.
- An **i.MX Audio Video Player** capable board. Note that a listener can be configured to play only audio or only video. In this latter case (no audio playout), an i.MX Video Renderer can be used.

Up to 10 listeners can be part of the evaluation setup.

- An **i.MX Video Renderer** capable board can be implemented as listener. The AVB configuration is described in the following section.
- An **i.MX Audio Amplifier** capable board can be implemented as listener. The AVB configuration is described in the following section.

- An AVB switch is required to interconnect the talker and listeners endpoints.

7.2 Setup preparation

The hardware preparation of the i.MX platforms is similar to the AVB Audio Media Server setup, with the variation that several listeners may be prepared, with or without screen display. For the video endpoints the HDMI port can be used for connecting a display.

All the endpoints should be connected to an AVB switch. The configuration of the AVB stack on talker and listeners is specific to this evaluation setup, and should be completed as described in the next sections.

7.2.1 Preparing the AVB configuration for the Talker

The default AVB script must be modified to configure operations of the Talker entity as using a custom Media Application. The AVB Stack is provided with a Media Server application example, interfaced to the AVB stack through the GenAVB/TSN API, and supporting reception of A/V samples from GStreamer reading from media files. To enable this media application, the GenAVB/TSN avb configuration file should be modified as follows:

1. Power on the i.MX board and let the boot process complete.
2. Edit the `GenAVB/TSN avb` configuration file using the following command at the Linux prompt:

```
# vi /etc/genavb/config_avb
```

3. Set the configuration profile to PROFILE 4:

```
PROFILE=4
```

4. Make sure that the `GenAVB/TSN systemd` service is enabled to start at boot:

```
# systemctl enable genavb-tsn
# systemctl daemon-reload
```

Enable the system clock to be gPTP-based:

```
CFG_USE_PHC2SYS=1
```

Exit and save the file.

5. The A/V file to be streamed should be stored in the `/home/media` repository. Some A/V files are available in the /home/media repository. They are named as follows:

```
# ls /home/media
sintel_trailer_720p_TOVERLAY.mp4
```

The full path to the A/V file, or at least a directory containing a list of mp4 files, should be mentioned as input parameter to the Media Application. For this purpose, the GenAVB/TSN profile should be modified as follows:

6. Edit the `GenAVB/TSN` video talker profile using the following command at the Linux prompt:

```
# vi /etc/genavb/apps-talker-video.cfg
```

The media application to be used is specified as follows:

```
CFG_EXTERNAL_MEDIA_APP="genavb-media-app"
```

The media application option string contains the below application options:

```
CFG_EXTERNAL_MEDIA_APP_OPT='${CFG_MEDIA_CLOCK} -T -A 0 -m 0:1
 -T -A 1 -m 0:0 -T -A 2 -m 0:2 -f /home/media/'
```

The option string indicates that the media application is configured for multitalker streams pipeline instance **0** (-m **0**:x) and each AVDECC stream index (-A <X>) is mapped to the right pipeline sink (-m 0:**y**). The list of all the multitalker streams pipeline instances can be showed by visualizing the application help (`$genavb-media-app -h`). Each talker stream configuration is separated and preceded by the `-T` option.

7. The media application option string can be extended/modified by the following arguments, preferably at the end of the string (to not be overridden):
   - '`-f`' indicates the path to the A/V file to be read or the path to an `mp4` media files directory
   - '`-l`' enables video local preview (on the talker)
   - '`-d`' indicates the display type to be used (`lvds` or `hdmi`)
   - '`-p`' <delay> sets the latency (in ns) added to the local preview rendering
8. Exit and save the file. Reboot the board.
   The change is saved across reboots so this has only to be done once.

Note: *While configuring the Talker with local preview rendering, a screen should be connected to the board. Otherwise, the GStreamer pipeline would not perform video looping.*

7.2.2 Preparing the AVB configuration for the Listeners

The default AVB script needs to be modified to configure operations of the Listener entity as using a custom Media Application. The AVB Stack is provided with a video application, interfaced to the AVB stack through the GenAVB/TSN API, and supporting A/V playout to GStreamer. To enable this media application, the GenAVB/TSN avb configuration file needs to be modified as follows:

1. Power on the i.MX board and let the boot process complete.
2. Edit the GenAVB/TSN avb configuration file using the following command at the Linux prompt:

```
# vi /etc/genavb/config_avb
```

3. For an Audio/Video listener, set the configuration profile to PROFILE 5:

```
PROFILE=5
```

4. For an Audio only listener, set the configuration profile to PROFILE 6:

```
PROFILE=6
```

5. For a Video only listener, set the configuration profile to PROFILE 16:

```
PROFILE=16
```

6. Make sure that the `GenAVB/TSN systemd` service is enabled to start at boot:

```
# systemctl enable genavb-tsn
# systemctl daemon-reload
```

7. Enable the system clock to be gPTP-based:

```
CFG_USE_PHC2SYS=1
```

8. Exit and save the file. The GStreamer options are controlled through the parameters passed to the Media Application.
   To change those parameters, the `GenAVB/TSN` profile should be modified as described in the next step.
9. For an Audio/Video listener, edit the video listener profile by using the following command at the Linux prompt:

```
# vi /etc/genavb/apps-listener-video.cfg
```

   - The Media Application to be used is specified as follows:

```
CFG_EXTERNAL_MEDIA_APP="genavb-media-app"
```

   - The Media Application option string contains GStreamer options:

```
CFG_EXTERNAL_MEDIA_APP_OPT='${CFG_MCR_SLAVE} -L -A 0 -g 0 -
p 305000000
 -a -v -d ${CFG_PRIMARY_VIDEO_DEVICE} -P
 ${CFG_ALSA_PLAYBACK_DEVICE}
-L -A 1 -g 1 -p 305000000 -v -d
 ${CFG_PRIMARY_VIDEO_DEVICE}'
```

     The *options* string indicates that the media application is configured for two AVDECC (-A <X>) streams. Each of the streams goes to a single GStreamer pipeline (-g <Y>) on connection time.
     - Stream index `0` is configured for an audio video stream that displays on the configured primary video device (HDMI or LVDS).
     - Stream index `1` is configured for a video stream that displays on the configured primary video device (HDMI or LVDS).

     Only the first connected AVDECC stream is displayed.
   - The Media Application option string can be extended/modified by the following arguments for each stream:
     - '-a' indicates to playout audio only from the IEC_61883_CIP_FMT_4 Stream.
     - '-v' indicates to playout video only from the IEC_61883_CIP_FMT_4 Stream.
     - '-d' indicates the display type to be used (lvds or hdmi).
     - '-p' <delay> sets the decoding delay (ns) for the local rendering.
10. The AVB options are specified in the second cfg file:

```
  # vi /etc/genavb/genavb-listener-video-btb.cfg
```

    The `talker_unique_id_list` indicates the AVDECC `talker_unique_ID` stream source. Its value is:
    - `0` for the RAW audio stream
    - `1` for the MPEG2-TS A/V stream
    - `2` for the CVF H264 Video stream

The `listener_unique_id_list` indicates the AVDECC `listener_unique_ID` stream sink (0 for the first stream).

11. For an Audio only listener, edit the audio listener profile by using the following command at the Linux prompt:

```
# vi /etc/genavb/apps-listener-audio.cfg
```

The Media Application to be used is specified as follows:

```
CFG_EXTERNAL_MEDIA_APP="genavb-media-app"
```

The Media Application option string contains GStreamer options:

```
CFG_EXTERNAL_MEDIA_APP_OPT='${CFG_MCR_SLAVE} -L -A 0 -g 0 -p
 305000000 -P ${CFG_ALSA_PLAYBACK_DEVICE}'
```

The options string indicates that the media application is configured for one AVDECC (-A 0) stream with index 0 that plays through a single GStreamer pipeline.

- '-p' <delay> sets the decoding delay (ns) for the local rendering.
- The AVB options are specified inside the second cfg file:

```
# vi /etc/genavb/genavb-listener-btb.cfg
```

- The '`talker_unique_id_list`' indicates the AVDECC `talker_unique_ID` stream sources:
  - `0` for the RAW audio stream.
  - `1` for the MPEG2-TS A/V stream.
  - `2` for the CVF H264 Video stream.
- '`listener_unique_id_list`' indicates the AVDECC `listener_unique_ID` stream sink (`0` for the first stream).
- For a Video only listener, edit the video listener profile by using the following command at the Linux prompt:

```
 # vi /etc/genavb/apps-listener-video.cfg
It uses the same configuration as for the Audio/Video
 Listener, except that the second pipeline will be
 launched.
```

The AVB options specified inside the second cfg file are:

```
  # vi /etc/genavb/genavb-listener-video-h264-btb.cfg
```

- '`talker_unique_id_list`' indicates the AVDECC `talker_unique_ID` stream source:
  - '`0`' for the RAW audio stream.
  - '`1`' for the MPEG2-TS A/V stream.
  - '`2`' for the CVF H264 Video stream.
- '`listener_unique_id_list`' indicates the AVDECC `listener_unique_ID` stream sink (0 for the first stream).

12. Exit and save the file, and then reboot the board. The change is saved across reboots so it needs to be done only once.
The setup is then ready for evaluation.

7.3 Evaluation instructions

On both talker and listener boards, power and boot the boards, and log in to the boards as root (no password).

The AVB stack starts automatically on each endpoint (as per `genavb-tsn` systemd service configuration described in Section 2.3.2).

The endpoints configuration is using the "*Fast Connect*" option, so the listeners automatically connect their input stream to the talker output streams once detected.

The Media Server runs a video talker application using GStreamer to read an A/V MP4 file from the SD flash storage encoded in H.264/AAC. The A/V source file is selected according to the file name given as input parameter to the talker application (see previous section about preparation of the talker configuration). The A/V file is read indefinitely in a loop.

From this audio/video muxed source, GStreamer pipelines are configured to generate an MPEG2-TS A/V stream, a RAW audio stream and another CVF H264 Stream. As soon as a stream is connected, the talker application starts reading the source media file and streaming samples to the AVB stack for transport to listeners. If any or all streams are connected, the corresponding samples are passed to the AVB stack. Hence three AVB class A streams can be generated by the talker:

- RAW audio 2 channel stereo 48 KHz in an IEC 61883-6 AVTP format
- MPEG2-TS audio/video AAC/H.264 in an IEC 61883-4 AVTP format.
- H264 video stream in a CVF H264 AVTP format.

Several listeners may listen to the same stream. Those streams are multicast to the connected listeners that are running the GStreamer application for A/V decoding and playout to the A/V devices. A/V is rendered on the display and/or speakers connected to the listener board, depending on the parameters passed to the listener application. Refer to the Section 7.2 for details about preparation of the listener configuration).

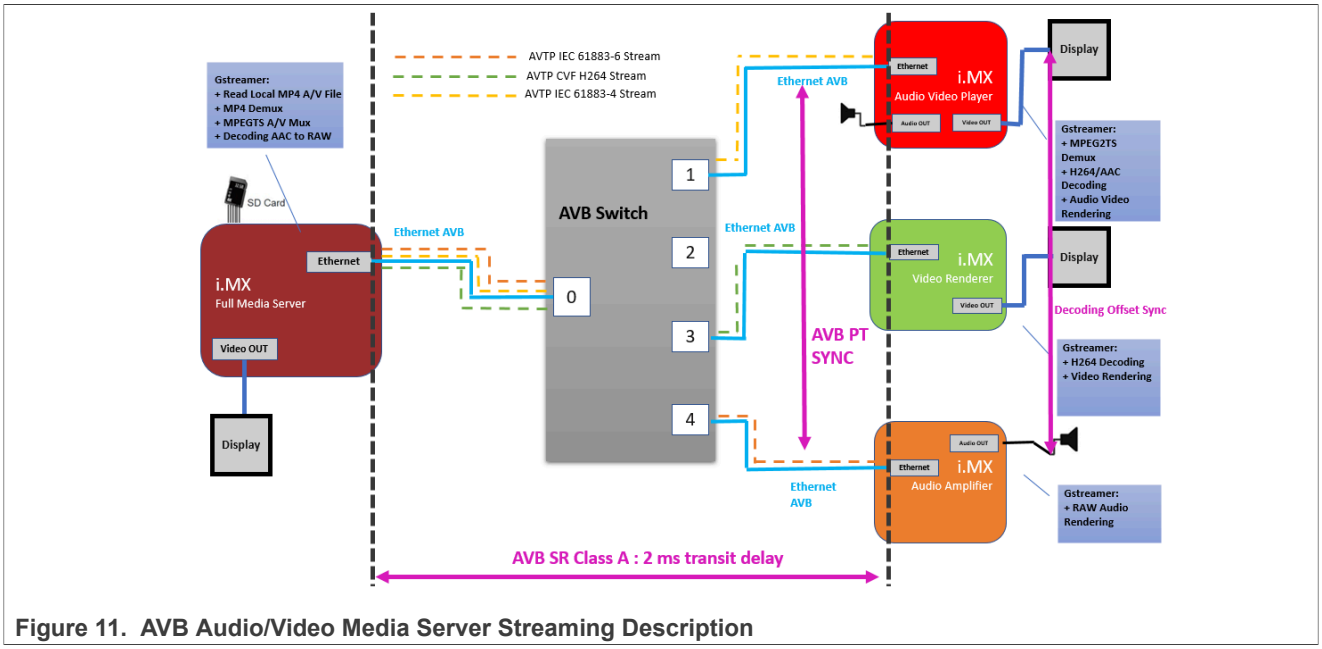

Figure 11. AVB Audio/Video Media Server Streaming Description

The evaluation can be stopped using the following command on either Talker or Listeners side:

```
# systemctl stop genavb-tsn
```

It can be re-started using the following command:

```
# systemctl start genavb-tsn
```

7.4 Audio/Video file format

The A/V file format is expected to be AAC/H.264 encoded, in an MP4 file container. The audio original format should be two channels with 48 KHz sampling rate.

8 AVB Media Synchronization Use Case

This section describes a use case for measuring media synchronization performance. It relies on using an AVB Audio live source stream from a talker transmitted to several listeners and running on three i.MX/Linux evaluation boards connected through an AVB switch.

8.1 Requirements

The setup requirement is as follows:

- The i.MX talker is reading samples from a live source (i.MX Audio Sampler role). The platform is similar to the talker used in the AVB Audio Sampler/Amplifier Back-to-back setup.
- The i.MX listeners are Audio Amplifiers similar to the listener used in the AVB Audio Sampler/Amplifier Back-to-Back setup. Up to four listeners can be part of the evaluation setup. (i.MX Audio Amplifiers role)
- An AVB switch is required to interconnect the talker and listeners nodes.

8.2 Setup preparation

- The hardware preparation of the i.MX evaluation platforms is similar to the AVB Audio Sampler/Amplifier Back-to-Back setup, with the variation that several listeners may be prepared. All those endpoints should be connected to an AVB switch. The configuration of the AVB stack on talker and listeners is specific to this evaluation and should be completed as described in the next sections.

8.2.1 Preparing the AVB configuration for the Talker

The default AVB script needs to be modified to configure operations of the Talker entity as using a custom Media Application. The AVB Stack is provided with an ALSA application, supporting audio sampling from an ALSA interface. To enable this media application, the GenAVB/TSN avb configuration file should be modified as follows:

1. Power on the i.MX evaluation board and let the boot process complete.

2. Edit the GenAVB/TSN avb configuration file using the following command at the Linux prompt:

```
# vi /etc/genavb/config_avb
```

3. Set the configuration profile to PROFILE 14:

```
PROFILE=14
```

4. Exit and save the file, then reboot the board.
5. The change is saved across reboots, so this process is required to be done only once.

The setup is then ready for evaluation.

8.2.2 Preparing the AVB configuration for the Listeners

The default AVB script needs to be modified to configure operations of Listeners entity as using a custom Media Application. The AVB Stack is provided with an ALSA application example, interfaced to the AVB stack through the GenAVB/TSN API, and supporting audio playout to an ALSA interface. To enable this media application, the GenAVB/TSN avb configuration file needs to be modified as follows:

1. Power on the i.MX evaluation boards and let the boot process complete.
2. Edit the GenAVB/TSN avb configuration file on each board using the following command at the Linux prompt:

```
# vi /etc/genavb/config_avb
```

3. Set the configuration profile to PROFILE 15:

```
PROFILE=15
```

4. Edit `/etc/genavb/genavb-audio-multi-btb-aaf.cfg` to set:

```
fast_connect = 0
btb_demo_mode = 0
```

5. Exit and save the file, then reboot the board. The change is saved across reboots, so this has only to be done once per board.

8.3 Evaluation instructions

On both talker and listeners boards, power on and boot the boards, and then log in it as root (no password). The AVB stack is automatically started on each endpoint (per `genavb-tsn` systemd service configuration described in Section 2.3.2).

To connect streams, use the command:

```
genavb-controller-app -c <talker_entity_id> 0
 <listener_entity_id> 0 0
```

To disconnect streams, use the command:

```
genavb-controller-app -d <talker_entity_id> 0
 <listener_entity_id> 0 0
```

The AVB audio stream connection is NOT automatically established. It is the controller's role to establish connections and request disconnections (for example, a talker's

reboot does not automatically disconnect listeners from their stream). When PTP is synchronized (see section 10.5) between the two systems, a media player on the laptop can be used to play audio to its line out interface. Audio is rendered on the headset of all listener boards.

8.4 Measuring performance

To perform the measurement between two endpoints, a PC with a stereo LINE-IN and an audio editing software is needed to be able to capture audio on two different channels and measure the time/sample difference between them:

- For synchronization measurements: one channel should come from each listener
- For end-to-end latency measurement: one channel should come from the live audio source feeding the talker and the other should come from the listener

For that, multiple audio adapters and cables are needed to perform the measurement on the evaluation boards. A setup examples with boards having audio jack input/output are shown in the following figures:

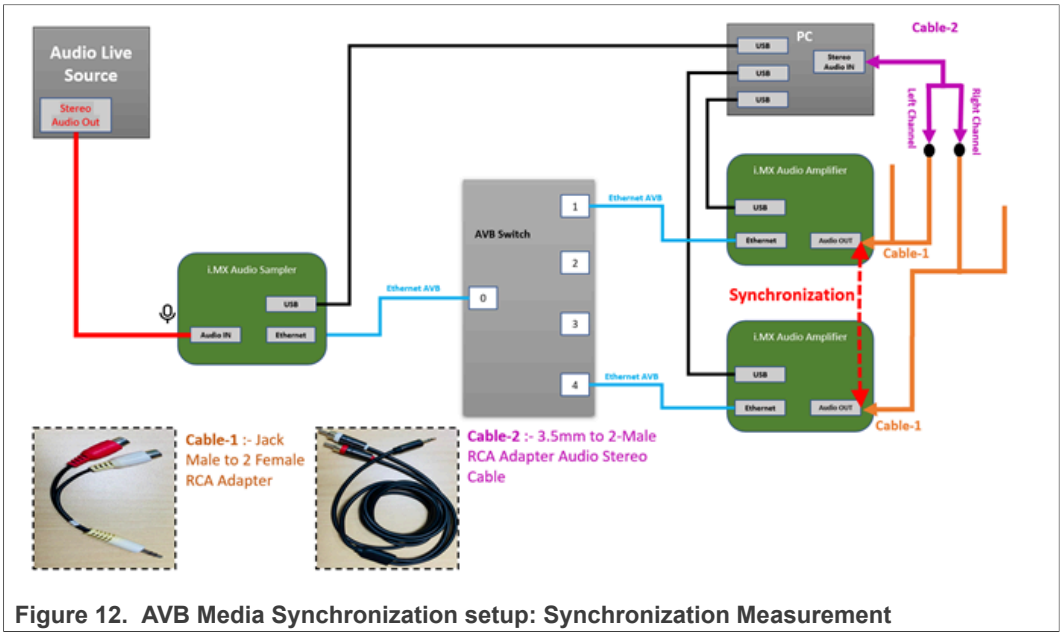

Figure 12. AVB Media Synchronization setup: Synchronization Measurement

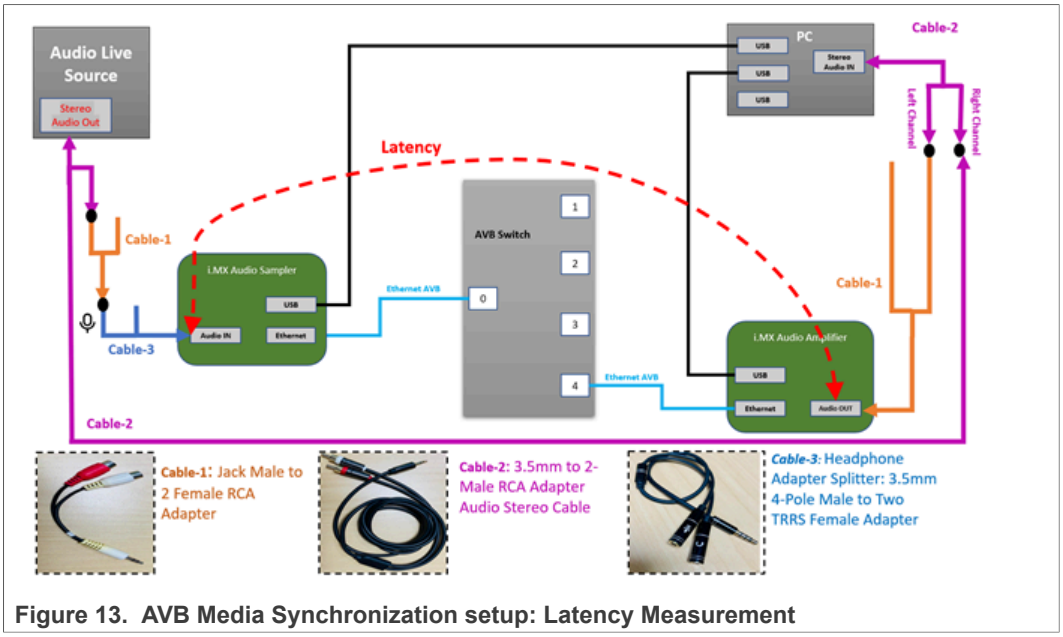

Figure 13. AVB Media Synchronization setup: Latency Measurement

The granularity of the measures is 20 µs for 1 sample. The synchronization performances should be below 50 µs between two listener boards (measured stable at 40 µs). The latency between the talker and one listener (end-to-end latency including application buffering) should be around 8.2 ms.

9 AVB Milan Audio Media Server/Amplifier

This section describes AVB Milan Audio Media Server and Amplifier, implemented on i.MX evaluation boards, connected through an AVB switch.

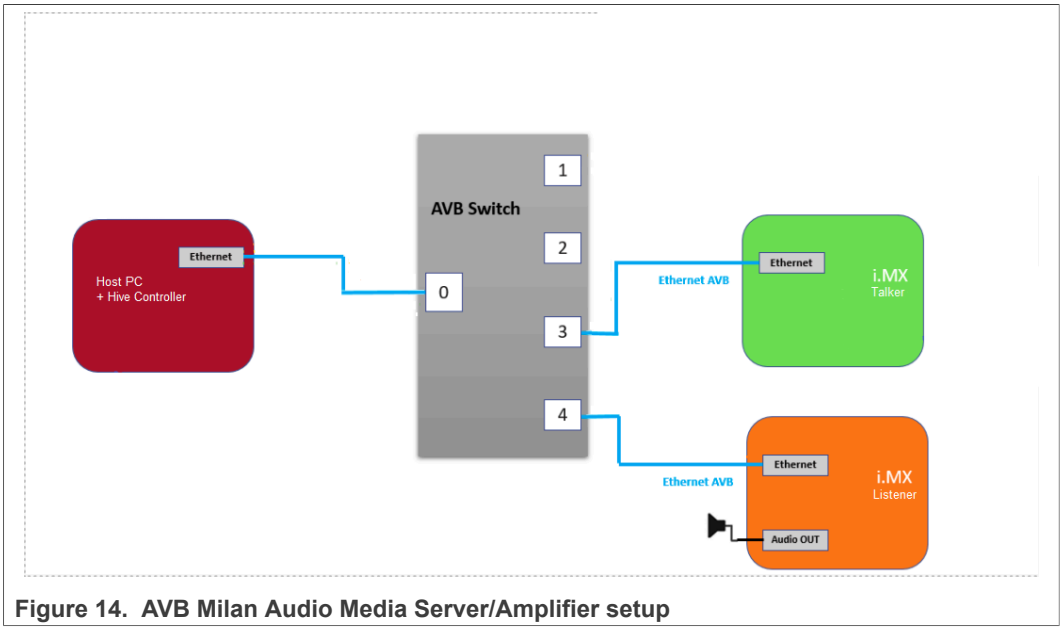

Figure 14. AVB Milan Audio Media Server/Amplifier setup

9.1 Requirements

The setup requirement is as follows:

- An **i.MX Full Media Server** capable board reading samples from files stored on the SD flash memory.
- An **i.MX Audio Amplifier** capable board can be implemented as listener. The AVB configuration is described in the following section.
- An AVB switch is required to interconnect the talker and listeners endpoints.
- A host PC with a Hive controller installed.

9.2 Setup preparation

The hardware preparation of the i.MX platforms is similar to the AVB Audio Media Server setup.

All the endpoints should be connected to an AVB switch. The configuration of the AVB stack on talker and listeners is specific to this evaluation setup and should be completed as described in the next sections.

9.2.1 Preparing the AVB configuration for the Talker

The default AVB script should be modified to configure operations of the Talker entity as using a custom Media Application. The AVB Stack is provided with a Media Server application example, which is interfaced to the AVB stack through the GenAVB/TSN API. The application supports reading audio samples from a media file. To enable using this media application, the GenAVB/TSN configuration file must be modified as follows:

1. Power on the i.MX board and let the boot process complete.
2. Edit the `GenAVB/TSN avb` configuration file using the following command at the Linux prompt:

```
# vi /etc/genavb/config_avb
```

3. Set the configuration profile to `PROFILE 20`:

```
PROFILE=20
```

4. Exit and save the file. Make sure that the GenAVB/TSN systemd service is enabled to start at boot:

```
# systemctl enable genavb-tsn
# systemctl daemon-reload
```

The raw audio file to be streamed should be stored in the `/home/media` repository. A sample Audio file is already available in the `/home/media/` directory. It is named as follows:

```
# ls /home/media
sample1_for_aaf.raw
```

The full path to the raw audio file, should be mentioned as input parameter to the Media Application. For this purpose, the `GenAVB/TSN` profile should be modified as follows:

5. Edit the GenAVB/TSN video talker profile using the following command at the Linux prompt:

```
# vi /etc/genavb/apps-talker-simple-aaf.cfg
```

6. The Media Application to be used is specified as follows:

```
CFG_EXTERNAL_MEDIA_APP="simple-audio-app"
```

   - The Media Application option string contains application options:

```
CFG_EXTERNAL_MEDIA_APP_OPT="-f/home/media/
sample1_for_aaf.raw "
```

   - '-f' indicates the path to the raw Audio file (should be in the format: `Raw Audio, S32BE, 2 channels, 48 KHz`).

7. For more information on the options, please refer to the application help (`$ simple-audio-app -h`).
8. Exit and save the file.
9. Reboot the board. The change is saved across reboots so this has to be done only once.

9.2.2 Preparing the AVB configuration for the Listener

The default AVB script should be modified to configure operations of the Listener entity as using a custom Media Application. The AVB Stack is provided with a video application, interfaced to the AVB stack through the GenAVB/TSN API, and supporting audio playout to an ALSA interface. Details of the GenAVB/TSN avb configuration file are as follows:

1. Power on the i.MX board and let the boot process complete.
2. Edit the `GenAVB/TSN avb` configuration file using the following command at the Linux prompt:

```
# vi /etc/genavb/config_avb
```

3. For an Audio/Video listener, set the configuration profile to PROFILE 21:

```
PROFILE=21
```

4. Exit and save the file.
5. Make sure that the `GenAVB/TSN systemd` service is enabled to start at boot:

```
# systemctl enable genavb-tsn
# systemctl daemon-reload
```

6. The alsa options are controlled through the parameters passed to the Media Application. To change those parameters, the GenAVB/TSN profile should be modified as follows:
   - For an Audio/Video listener, edit the video listener profile by using the following command at the Linux prompt:

```
# vi /etc/genavb/apps-listener-alsa-milan.cfg
```

   - The Media application to be used is specified as follows:

```
CFG_EXTERNAL_MEDIA_APP="alsa-audio-app"
```

   - The Media application option string contains the following:

```
CFG_EXTERNAL_MEDIA_APP_OPT="-d ${CFG_ALSA_PLAYBACK_DEVICE}
 -b /etc/genavb/milan_binding_params.nvram"
```

- '-b' indicates the file location where binding parameters must be saved in non-volatile memory (as per Milan specification), so that streaming can be recovered without controller intervention in case of a power cycle or network disruption or a similar situation.

7. Exit and save the file, and then reboot the board. The change is saved across reboots so it needs to be done only once. The setup is then ready for evaluation.

9.3 Evaluation instructions

On both talker and listener boards, power and boot the boards, and log in to the boards as root (no password).

The AVB stack starts automatically on each endpoint (per `genavb-tsn` systemd service configuration described in Section 2.3).

Once the stack is started on each endpoint, you can open the **Hive Controller**. You should, then, see two Milan compatible entities detected.

To connect the Talker's stream outputs to the Listener's stream inputs, click on the corresponding box in the matrix of the 'Stream Based' section. (see).

Once the streams are successfully connected, the selected box turns green.

The binding information including the talker, listener streams, and controller id is saved in non-volatile memory on the listener side, so that streams already bound are recovered automatically on board reboot. To remove these parameters, an explicit unbind should be done on the controller (or an explicit removal of the binding file on board).

9.4 Audio file format and generation

The current Media Server application uses a raw audio format:

- Two channels
- 48 kHz
- 32 bits
- Big endian numbering format

An i.MX Audio Sampler capable board can be used to generate such a file, by connecting an analog source (external player) to a Jack audio input port. The following ALSA command allows capturing and storing the file in the expected format:

```
# arecord_-D <device> -t raw -c 2 -r 48000 -f S32_BE <filename>
```

To verify and listen to the captured file, connect a hearing device (headphones/speakers) to audio output Jack port or MIC. (In case of the SABRE-AI board, use the RCA/Jack cable connected to Audio Line OUT port.)

Use the following ALSA command:

```
# aplay -D <device> -c 2 -r 48000 -f S32_BE <file name>
```

The alsa device should be assigned to audio codec connected to jack interface (wm8960-audio).

10 Preparing the Hive Controller on the Host PC

On the Host PC, you need to retrieve Hive Controller's binaries. It is recommended to use a Host PC under Windows/MacOS. Otherwise, you would need to compile the Hive Controller yourself.

Follow the instructions provided in the above link and execute the Hive Controller. Here is an example of the interface you would get, once both endpoints are started with the AVB stack running.

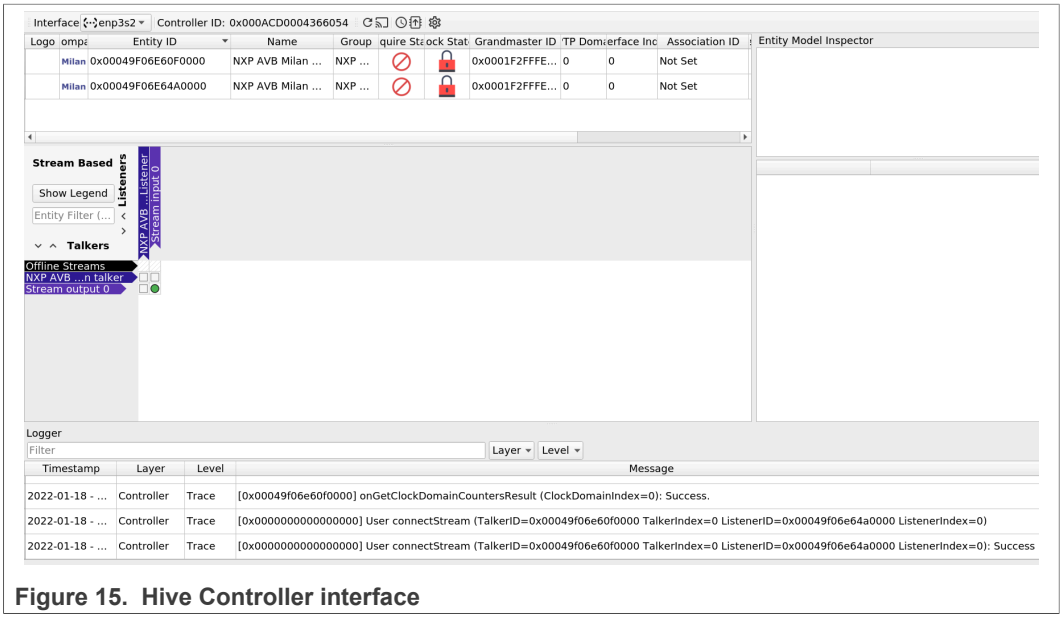

Figure 15. Hive Controller interface

11 AVB Milan Audio Sampler/Amplifier with CRF

This section describes AVB Milan Audio Sampler and Amplifier with CRF support, implemented on i.MX evaluation boards, connected through an AVB switch.

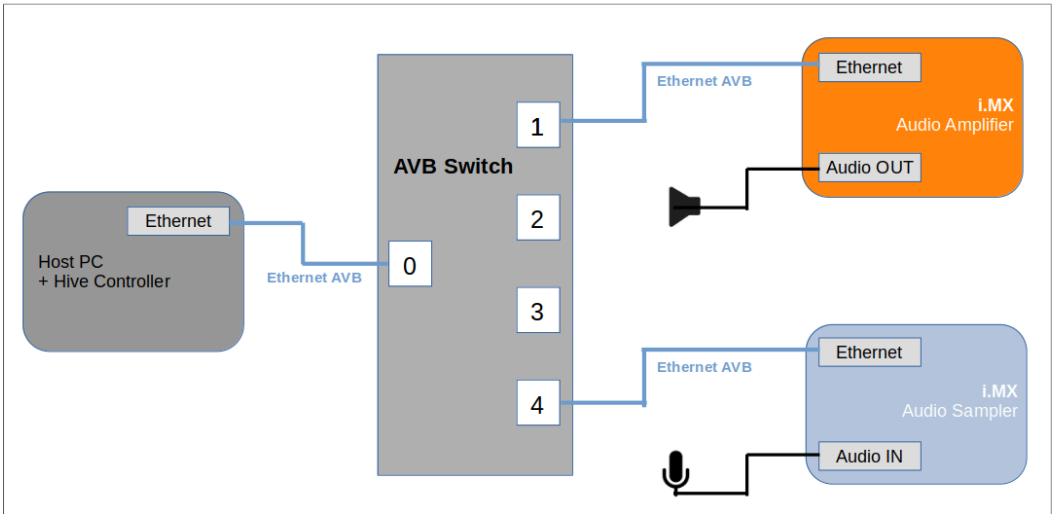

Figure 16. AVB Milan Audio Sampler/Amplifier with CRF setup

11.1 Requirements

The setup requirement is as follows:

- An **i.MX Audio Sampler** capable board with media clock recovery support (for example, i.MX8MP or i.MX6ULL with hardware rework).
- An **i.MX Audio Amplifier** capable board with media clock recovery support (for example, i.MX8MP or i.MX6ULL with hardware rework).
- An AVB switch is required to interconnect the talker and listeners endpoints.
- A host PC with a Hive controller installed.
- Headphones/speakers with male Jack.

11.2 Setup preparation

The hardware preparation of the i.MX platforms is similar to the AVB Audio Sampler/ Amplifier setup. All the endpoints should be connected to an AVB switch. The configuration of the AVB stack on talker and listeners is specific to this evaluation setup and should be completed as described in the next sections.

11.2.1 Preparing the AVB configuration for the Talker

The default AVB script should be modified to configure operations of the Talker entity for use as a custom Media application. The AVB Stack is provided with a Media application example, interfaced to the AVB stack through the GenAVB/TSN API, and supports audio sampling from a speaker. To enable using this media application, the GenAVB/TSN avb configuration file should be modified as follows:

1. Power on the i.MX board and let the boot process complete.
2. Edit the `GenAVB/TSN avb` configuration file using the following command at the Linux prompt:

```
# vi /etc/genavb/config_avb
```

3. Set the configuration profile to profile 22:

```
PROFILE=22
```

4. Exit and save the file.
5. Reboot the board. The change is saved across reboots, so this has to be done only once.

11.2.2 Preparing the AVB configuration for the Listener

The default AVB script needs to be modified to configure operations of the Listener entity as using a custom Media Application. The AVB Stack is provided with a Media application example, interfaced to the AVB stack through the GenAVB/TSN API, and supporting audio playout from a headphone. To enable using this media application, the `GenAVB/TSN avb` configuration file needs to be modified as follows:

0. Power ON the i.MX board and let the boot process complete.

1. Edit the `GenAVB/TSN avb` configuration file using the following command at the Linux prompt:

```
# vi /etc/genavb/config_avb
```

2. Set the configuration profile to profile 22:

```
PROFILE=22
```

3. Exit and save the file.

4. Reboot the board. The change is saved across reboots, so this has to be done only once.

11.2.3 Preparing the Hive Controller on Host PC

On the Host PC, you need to retrieve Hive Controller's binaries and follow the instructions to compile and install it depending on your platform.

Then, follow the link's instructions to execute the Hive Controller.

Here is an example of the interface you would get once both endpoints have started with the AVB stack running.

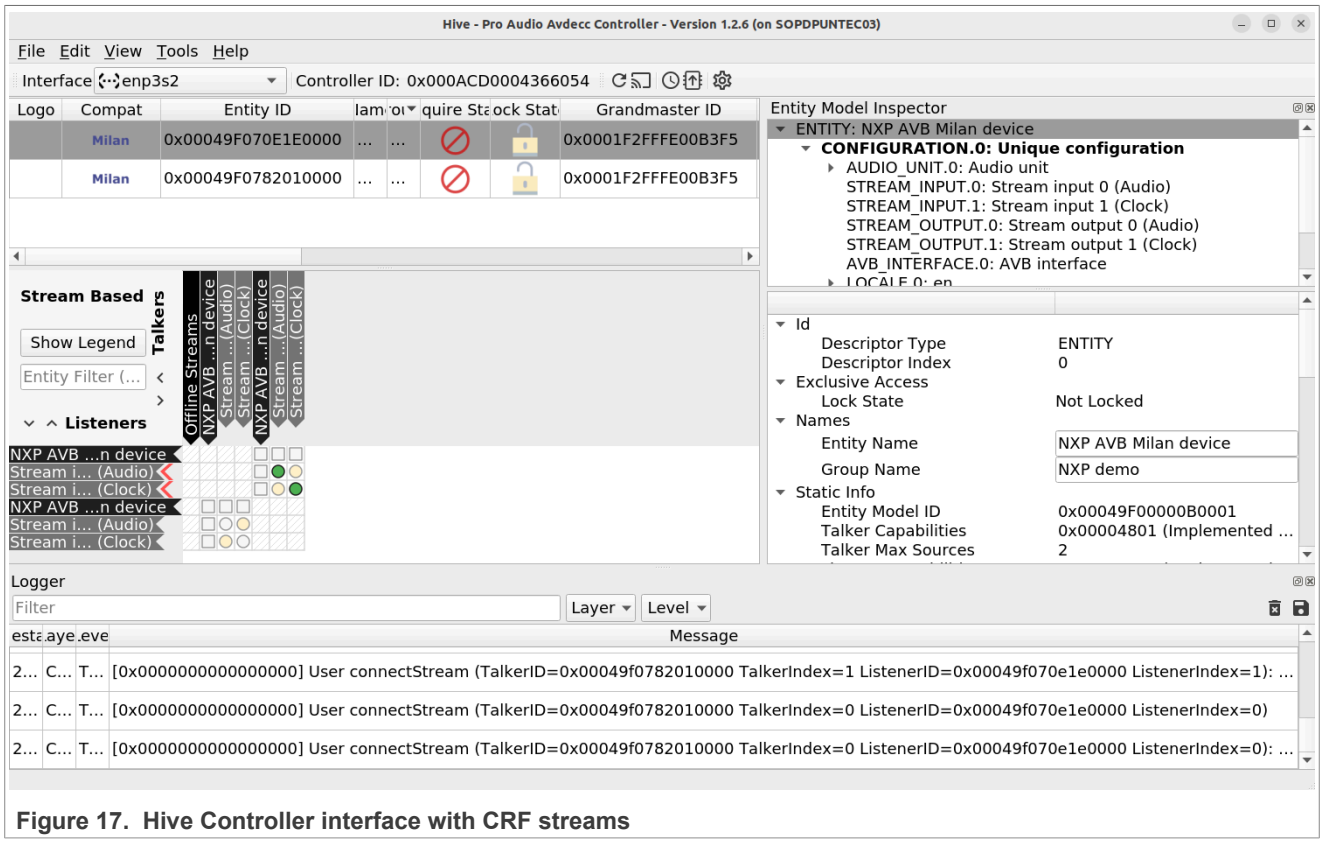

Figure 17. Hive Controller interface with CRF streams

11.3 Evaluation instructions

On both endpoints, power and boot the boards, and log in as root (no password).

The AVB stack starts automatically on each endpoint (per `genavb-tsn` systemd service configuration described in Section 2.3.2).

Once the stack is started on each endpoint, you can start the Hive Controller on the Host PC.

Two Milan compatible entities should be detected and appear on the interface.

To connect a stream, you need to click on the corresponding box in the matrix of the 'Stream Based' section. (See Section 11.2.3). Once the streams are successfully connected, the selected box turns green.

- Connect the CRF stream input(s) 'Stream input 1 (Clock)' of the endpoint(s) that supports media clock recovery and that would act as the CRF Slave(s) to the CRF stream output 'Stream output 1 (Clock)' of the endpoint that would act as the CRF master.

Once the connection is reported successful (green dot)

- Connect the Audio stream input 'Stream input 0 (Audio)'of the endpoint you want to use as a Listener to the Audio stream output 'Stream output 0 (Audio)' of the endpoint you want to use as a Talker.

Once the connection is reported successful (green dot), you can hear on the Listener's headphone, any sound passed into the Talker's audio input port.

12 More on evaluation usage

This section provides more information on the GenAVB/TSN evaluation package usage.

12.1 AVB stack start/stop

By default, the GenAVB/TSN stack is not started on boot. But, a `systemd` service is available to enable that option. The service file can be found at `/lib/systemd/system/genavb-tsn.service`. It can be disabled/re-enabled by using the following commands:

```
# systemctl <enable|disable> genavb-tsn
```

The AVB stack can be manually started and stopped at any time using the following command:

```
# systemctl start genavb-tsn
```

It loads the kernel modules, starts the TSN daemon if it is not running, sets up device nodes and processes priorities, and starts both the AVB daemon and demo application.

```
# systemctl stop genavb-tsn
```

It stops the AVB application but keeps the TSN daemon (gPTP and SRP stacks) running (so that PTP synchronization can be kept even if the AVB stack is restarted).

Note:

1. *The gPTP and SRP stacks can be also started and stopped independently using the following command:*

```
# tsn.sh <start|stop>
```

2. *An option to stop/restart all the stack's daemons and apps (AVB and TSN) can be done with the following command:*

```
# avb.sh <stop_all|restart_all>
```

3. *Executing `avb.sh <start|stop>` can still be used to start/stop the stack with the same manner. However, it is recommended to use `systemctl` to keep `systemd` updated about the status of the service, as shown below:*

```
# systemctl status genavb-tsn
```

4. *The logs from the GenAVB/TSN service can be found in the `systemd` journal by executing the command below:*

```
# journalctl
```

12.2 AVB stack logs

More information about the stack behavior may be found there, in particular about the playback delays. Logs are stored in `/var/log/avb`. This is a `tmpfs` that is lost on reboot.

- Linux command:

```
# tail -f /var/log/avb
```

- Focus on AVB statistics displayed every 10 seconds in the log file:

```
# tail -f /var/log/avb | grep "stats_print"
```

- Meaning of media stack statistics:

```
avtp stream_listener_stats_print : now-rx_ts   37/  58/  95
```

- Min/average/max in µs for the difference between the time the packet was received by the media stack and the time the packet was timestamped by the ENET on the receiver.

```
avtp stream_listener_stats_print : avtp_ts-now 1772/1809/1836
```

- Min/average/max in µs for the difference between the AVTP timestamp inside the packet and the time the packet was received by the media stack.

12.3 SRP stack logs

SRP stack is running inside the TSN process. Logs for SRP stack are stored in `/var/log/tsn` . This is a `tmpfs` that is lost on reboot.

- Linux command:

```
# tail -f /var/log/tsn | grep srp
```

12.4 AVB applications logs

More information about the running media application may be found there. Logs are stored in `/var/log/avb_media_app`. This is a `tmpfs` that is lost on reboot.

- Linux command:

```
# tail -f /var/log/avb_media_app
```

Displayed statistics are dependent on the application.

- Meaning of media stack statistics:

```
alsa latency  895/1020/1187
```

The above command displays min/average/max in µs for the playback delay of frames sent to the ALSA stack.

12.5 gPTP endpoint statistics

The gPTP stack runs inside the TSN process. Logs for gPTP are stored in `/var/log/tsn` , but a gPTP filtered log file is available in `/var/log/fgptp`. This is a `tmpfs` that is lost on reboot.

- Linux command:

```
# tail -f /var/log/fgptp
```

- If the stack is configured in automotive mode, then the log contains:

```
Running fgptp in automotive profile on interface eth0
```

- *Port Role*, *Port AS-capability*, *link Status*, *neighbor capability* and *delay mechanism* are reported each time there is a change in the link state (link is 802.1AS capable or not) or upon Grand Master (GM) change. This information is also displayed regularly along with current synchronization and pdelay statistics

```
Port(0): domain(0,0): role changed from DISABLED to SLAVE
…
Port(0): domain(0,0): Slave - Link: Up - asCapable: Yes
 neighborGptpCapable: Yes delayMechanism: P2P
```

- Selected Grand Master (GM) capabilities are reported upon new GM selection. *Root Identity* represents the clock ID of the currently selected GM. *Priority1, Priority2, Class* and *Accuracy* describe the clock quality of the selected GM. Finally, the *Peer Master Identity* of the nearer peer master endpoint the current device is connected to (for example, a switch placed between the slave device and the GM):

```
Grand master: root identity 00049ffffe039e35
Grand master: priority1 245 priority2
Grand master: class 248 accuracy 248
Grand master: variance 17258
```

```
Grand master: source port identity 0001f2fffe0025fe, port
 number 2
```

- *Synchronization State* is reported upon GM selection (SYNCHRONIZED) or when no GM are detected (NOT SYNCHRONIZED). Synchronization *Time* expressed in ms represents the time it took to the local clock to reach synchronization threshold starting from the first SYNC message received.

```
Port(0) SYNCHRONIZED - synchronization time (ms): 250
```

- *Pdelay* (propagation delay) and local clock adjustments are printed out every 5 seconds. *PDelay* is expressed in ns units and represents the one-way delay from the endpoint and its peer master. *Correction* is expressed in parts per billion and represents the frequency adjustment performed to the local clock. *Offset* is expressed in ns. It represents the resulting difference between the locally adjusted clock and the reference gPTP Master's clock. (Min/Max/Avg and Variance are computed for both Correction and Offset statistics)

```
Port(0): domain(0,0): Propagation delay (ns): 37.60  min 34
 avg 36 max 45 variance 17
Port(0): domain(0,0): Correction applied to local clock (ppb):
 min -5603 avg 5572 max 5538 variance 148
Port(0): domain(0,0): Offset between GM and local clock (ns)
 min -12 avg 4 max 22 variance 111
```

- The following port statistics (32 bits counters) are displayed every 15 seconds on slave and master entities:

Table 2. Port statistics (for 32 bits counters)

Receive counters	
`PortStatRxPkts`	Number of gPTP packets received (ether type 0x88F7)
`PortStatRxSyncCount`	Number of SYNC packets received
`PortStatRxSyncReceiptTimeouts`	Number of FOLLOW-UP packets timeout
`PortStatRxFollowUpCount`	Number of FOLLOW-UP packets received
`PortStatRxAnnounce`	Number of ANNOUNCE packets received
`PortStatAnnounceReceiptTimeouts`	Number of ANNOUNCE packets timeout
`PortStatAnnounceReceiptDropped`	Number of ANNOUNCE packets dropped by the entity
`PortStatRxSignaling`	Number of SIGNALING packets received
`PortStatRxPdelayRequest`	Number of PDELAY REQUEST packets received
`PortStatRxPdelayResponse`	Number of PDELAY RESPONSE packets received
`PortStatPdelayAllowedLostResponses Exceeded`	Number of excess of allowed lost responses to PDELAY requests
`PortStatRxPdelayResponseFollowUp`	Number of PDELAY FOLLOW-UP packets received
`PortStatRxErrEtype`	Number of ether type errors (not 0x88F7)
`PortStatRxErrPortId`	Number or port ID errors
Transmit counters	
`PortStatTxPkts`	Number of gPTP packets transmitted
`PortStatTxSyncCount`	Number of SYNC packets transmitted

Table 2. Port statistics (for 32 bits counters)*...continued*

PortStatTxFollowUpCount	Number of FOLLOW-UP packets transmitted
PortStatTxAnnounce	Number of ANNOUNCE packets transmitted
PortStatTxSignaling	Number of SIGNALING packets transmitted
PortStatTxPdelayReques	Number of PDELAY REQUEST packets transmitted
PortStatTxPdelayResponse	Number of PDELAY RESPONSE packets transmitted
PortStatTxPdelayResponseFollowUp	Number of PDELAY FOLLOW-UP packets transmitted
PortStatTxErr	Number of transmit errors
PortStatTxErrAlloc	Number of transmit packets allocation errors
Miscellaneous counters	
PortStatAdjustOnSync	Number of adjustments performed upon SYNC received
PortStatMdPdelayReqSmReset	Number of resets of the PDELAY REQUEST state machine
PortStatMdSyncRcvSmReset	Number of resets of the SYNC RECEIVE state machine
PortStatHwTsRequest	Number of egress timestamp requests
PortStatHwTsHandler	Number of egress timestamp notifications
PortStatNumSynchronizationLoss	Number or synchronization loss on the slave endpoint (for example, GM change, GM reference clock discontinuity, and so on)
PortStatNumNotAsCapable	Number of transitions from AS_Capable=TRUE to AS_Capable= FALSE

12.6 gPTP bridge statistics

gPTP stack is running inside the TSN process. Logs for gPTP are stored in `/var/log/tsn-br` , but a gPTP filtered log file is available in `/var/log/fgptp-br`.

This is a `tmpfs` that will be lost on reboot.

- Linux command:

```
# tail -f /var/log/fgptp-br
```

- The bridge stack statistics are similar to the endpoint stack ones except that they are reported for each of the external ports of the switch (Port 0 to 3) and also for the internal port connected to the endpoint stack (Port 4) in case of Hybrid setup.
- *Pdelay* (propagation delay), *Link status*, *AS capability*, *Port Role*, *neighbor capability* and *delay mechanism* are printed out for each port.

```
Port(0): Role: Disabled    Link: Up asCapable: No
 neighborGptpCapable: No delayMechanism: P2P
```

```
Port(1): Role: Disabled    Link: Up asCapable: No
 neighborGptpCapable: No delayMechanism: P2P
```

```
Port(2): domain(0,0): Role: Disabled    Link: Up asCapable: No
 neighborGptpCapable: Yes delayMechanism: P2P

Port(2): Propagation delay (ns): 433.98 min 425 avg 438 max 457
 variance 87
```

```
Port(3): domain(0,0): Role: Disabled    Link: Up asCapable: No
 neighborGptpCapable: No delayMechanism: P2P
```

```
Port(4): domain(0,0): Role Master    Link: Up    asCapable: Yes
 neighborGptpCapable: Yes delayMechanism: P2P

Port(4): Propagation delay (ns): 433.98 min 425 avg 438 max 457
 variance 87
```

12.7 Using the GenAVB/TSN AVDECC Controller

The evaluation package includes an AVDECC controller demo application. This application can be invoked from the Linux command prompt of any endpoint running the GenAVB/TSN stack:

```
# genavb-controller-app -h

Usage:
app [options]
Options:
-v <entity_id> <control_index> <value>
Set a given control to the given value (control must be of type
 UINT8)
-l
List discovered AVDECC entities
-c <talker_entity_id> <talker_unique_id> <listener_entity_id>
 <listener_unique_id> <flags>
Connect a stream between a talker and a listener
-d <talker_entity_id> <talker_unique_id> <listener_entity_id>
 <listener_unique_id>
Disconnect a stream between a talker and a listener
-r <listener_entity_id> <listener_unique_id>
Get information about a listener sink
-t <talker_entity_id> <talker_unique_id>
Get information about a talker source
-s <talker_entity_id> <talker_unique_id> <index>
Get information from a talker about a given connection/stream
-T <talker_entity_id> <talker_unique_id> <start|stop>
Send START_STREAMING or STOP_STREAMING command to a talker
-L <listener_entity_id> <listener_unique_id> <start|stop>

Send START_STREAMING or STOP_STREAMING command to a listener
-h
Print this help text
```

The “`-l`” option lists information and characteristics of all the AVDECC entities declared by the AVB endpoints present on the network. This information is used to control AVB

activity among the endpoints using other available options as described in the help output.

Logs from the AVDECC controller application can be displayed by using the following command:

```
# tail -f /var/log/avb_avdecc_controller
```

12.8 Net AVB kernel module statistics

The module exports multiple stats through debugfs under `/sys/kernel/debug/avb/`

- Software FQTSS and TX stats: Stats about the different TX streams/traffic classes and the software FQTSS used in the net AVB kernel module are shown under `/sys/kernel/debug/avb/tx/`
- RX stats: Stats about received packets in the net AVB kernel module with their different protocols are shown under: `/sys/kernel/debug/avb/rx/`
- Hardware timer stats: Stats about the hardware timer used to drive the packets' processing in the net AVB kernel module are shown under: `/sys/kernel/debug/avb/hw_timer/`
- Media clock stats: Stats about the medial clock driver in the net AVB kernel module are shown under: `/sys/kernel/debug/avb/mclock/`

When the AVB endpoint is configured as media clock slave with enabled media clock recovery, the stats are shown under: `/sys/kernel/debug/avb/mclock/rec_pll_0`

```
# cat /sys/kernel/debug/avb/mclock/rec_pll_0
```

When the recovery mechanism is running, the stats should show the number of adjustment "adjust" increasing on multiple reads and should print the drift between the audio clock (slave clock) and the master clock.

The latest applied adjustment (in ppb) to the audio PLL is shown under the field: "`last applied ppb adjust`"

12.9 Setting a static IP address for the network interface

The system uses *systemd-networkd* service to manage network configurations.

To configure the network interfaces (either as DHCP client or static IP address), a network configuration file can be added at `/etc/systemd/network/70-eth0.network`.

To configure the eth0 interface with a static IP address, set the desired IP address as in the following example:

```
[Match]
Name=eth0
[Network]
Address=192.168.1.5/16
Gateway=192.168.1.1
```

Then, restart the systemd-networkd service for the new configuration to take effect:

```
# systemctl restart systemd-networkd.service
```

The changes are saved across reboots.

13 Revision history

This table summarizes the changes made to this document.

Table 3. Document revision history

Revision number	Date	Changes
1	16 December 2022	Updated for Real Time Edge Software Rev 2.4 release
0	28 July 2022	Initial version for Real Time Edge Software Rev 2.3 release

14 Legal information

14.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

14.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

14.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

i.MX — is a trademark of NXP B.V.

Contents

For more information, please visit: http://www.nxp.com

Date of release: 16 December 2022