Ethernet AVB

Training Guide Version 1

Intel Confidential
Document Number: 603429, Revision 0.4

Internet of Things Group

Legal Disclaimer

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: http://www.intel.com/design/literature.htm

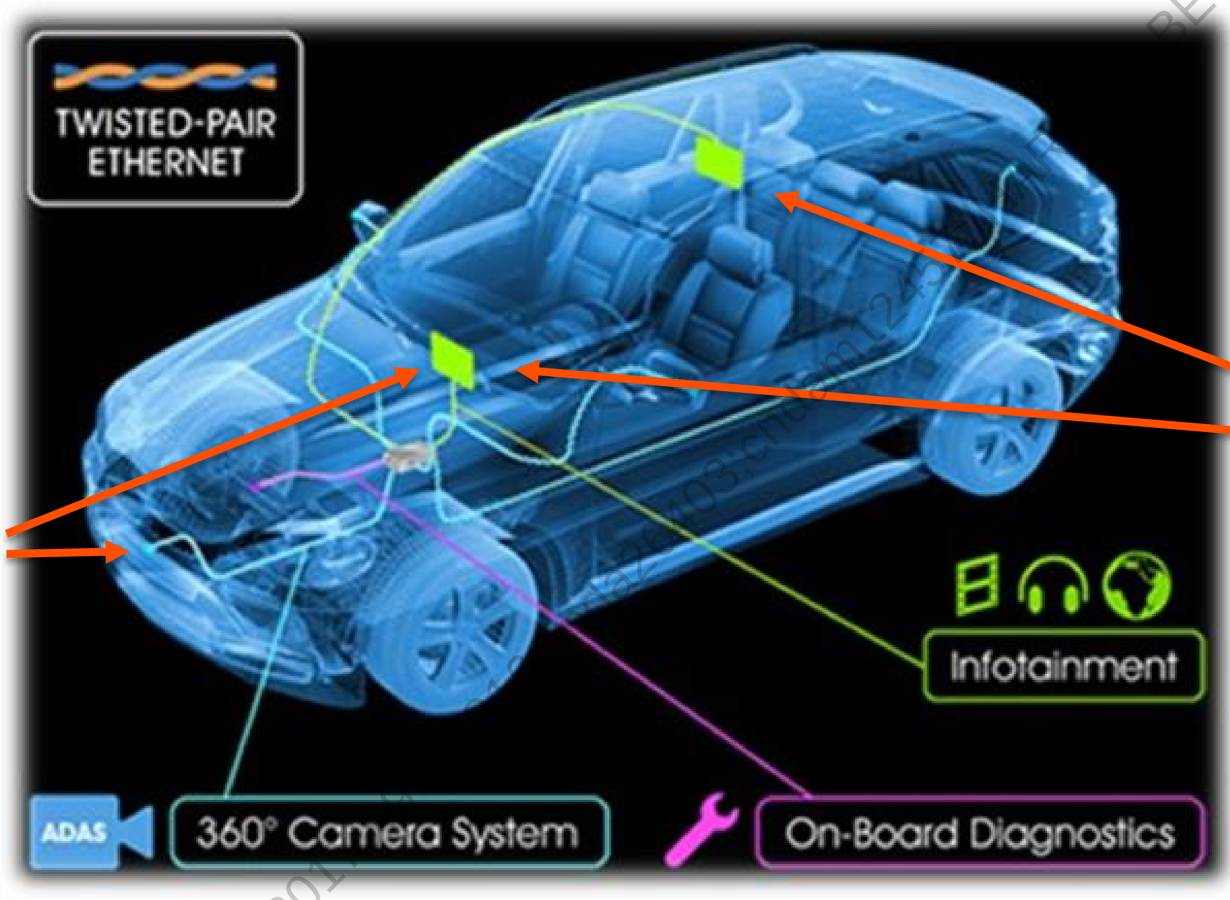
Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Introduction

Introduction

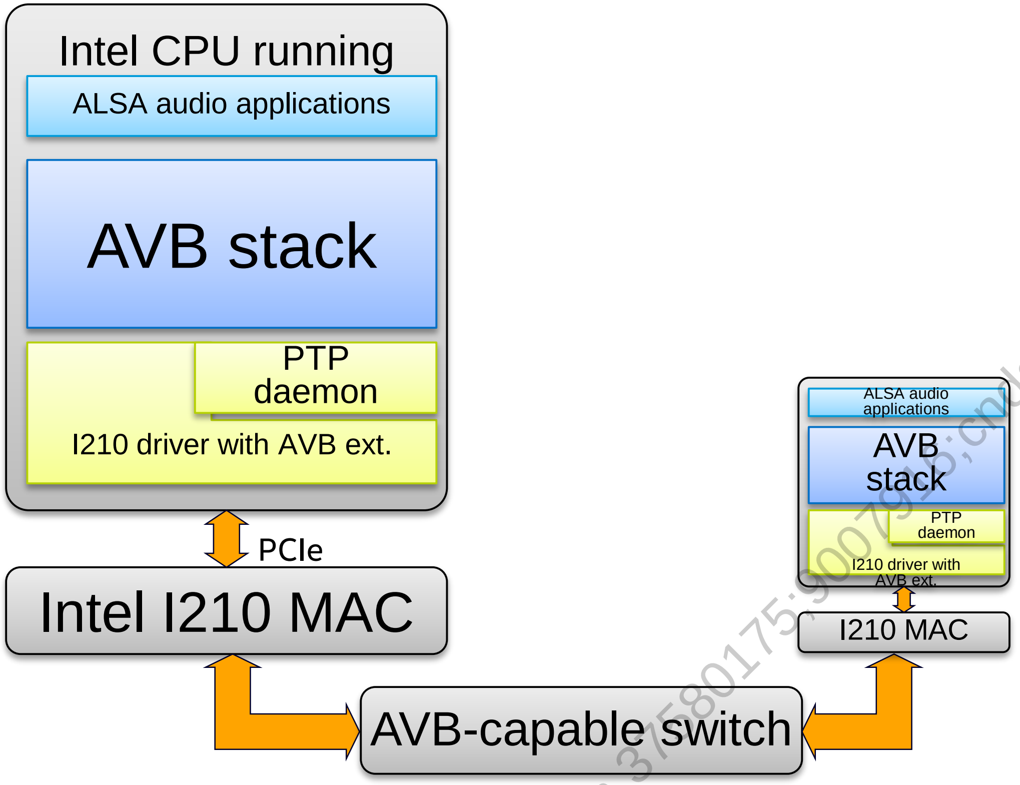
Video transmission between camera and front panel



Audio transmission between front and rear panel

Overview

Intel is a significant contributor to the **IEEE standards** related to AVB and a founding promoter of the **AVnu Alliance**. Intel is a maintainer of the **OpenAVB** project, using I210 under Linux, too.



Intel CPU running

ALSA audio applications

AVB stack

PTP daemon

I210 driver with AVB ext.

PCIe

Intel I210 MAC

AVB-capable switch

ALSA audio applications

AVB stack

PTP daemon

I210 driver with AVB ext.

I210 MAC

- Our Linux-based, automotive qualified AVB stack benefits from I210 Ethernet controller time synchronization and implements IEEE1722-rev1, 802.1Q class A/B/C, AVnu Automotive Profile.
- Our AVB transmission schemes support both, latency sensitive audio and video transmission – esp. for ADAS cameras and IVI media data, and early start-up with suspend/resume support!
- At the latest AVnu plug fest our AVB solution outperformed against reference devices from other vendors.

AVB StreamHandler

The AVB StreamHandler acts as link between the AVB-capable Ethernet* interface and audio/video applications local to the device.

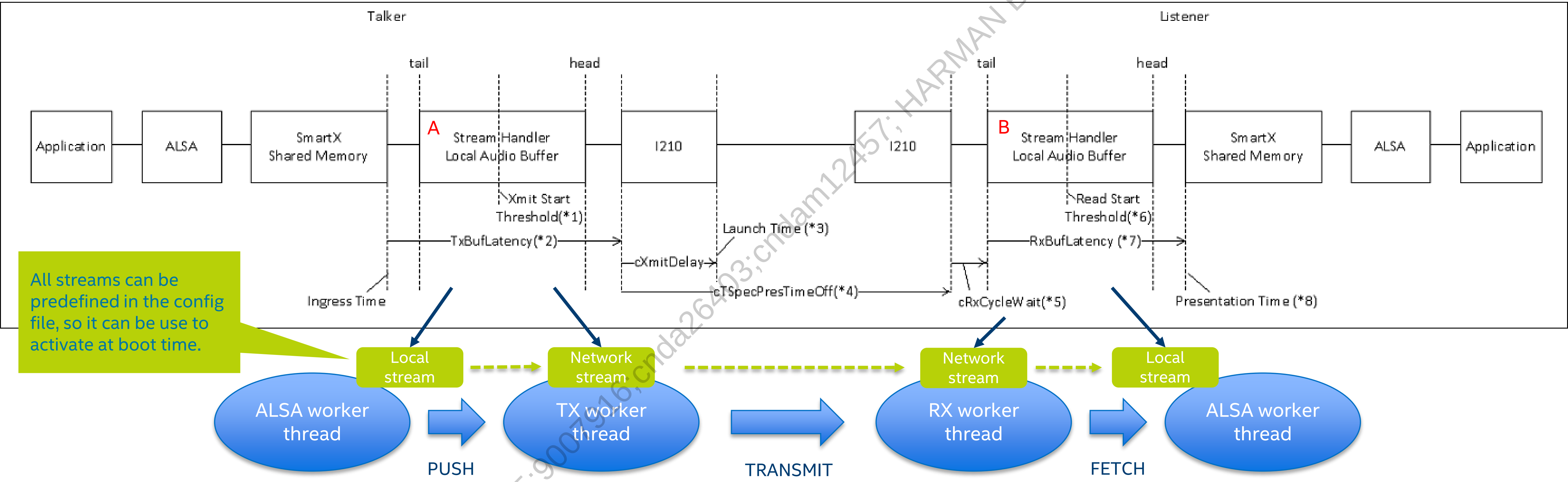
Audio applications can receive and transmit audio data through ALSA interfaces. Video applications can receive and transmit either H.264 encoded video or MPEG-TS streams via a ufipc interface.

The AVB StreamHandler ensures proper AVB packet generation following the IEEE1722 standard and observes the subset of IEEE802.1Q deemed relevant for automotive applications.

The AVB StreamHandler relies on an implementation of the IEEE802.1AS standard (gPTP) to establish a network-wide reference time base.

Flow

AVB Transmission Diagram



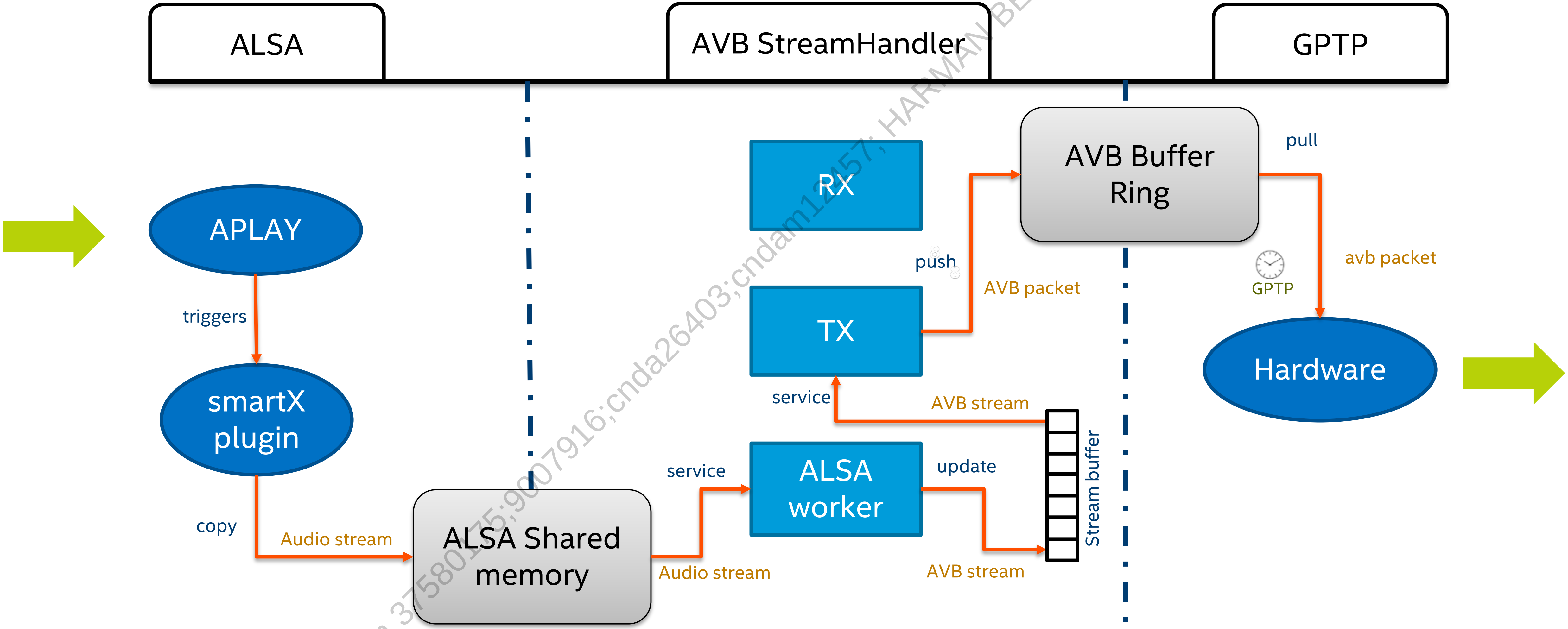
All streams can be predefined in the config file, so it can be use to activate at boot time.

Exceptional condition:

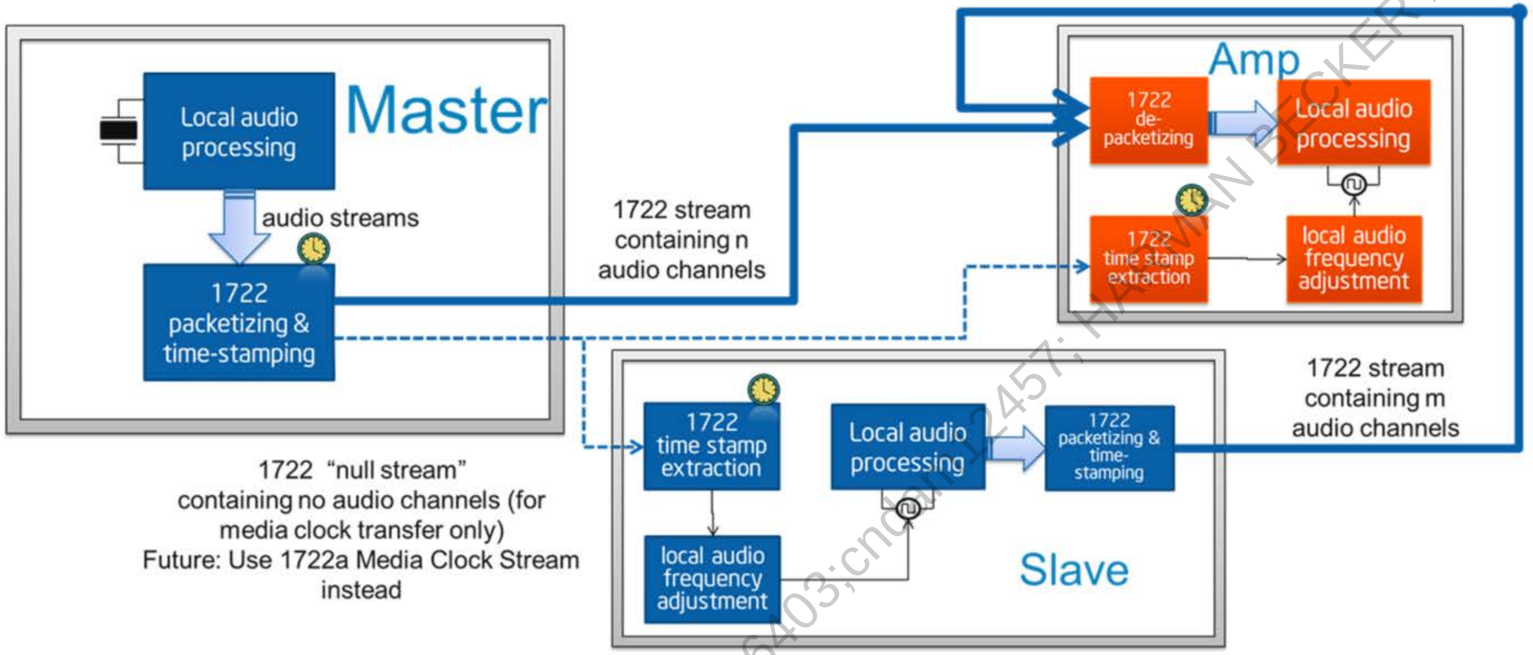
1. ALSA worker failed to push data, buffer A underrun.
2. ALSA worker failed to fetch data, buffer B overrun causes audio loss.

Flow – Thread and Process

Aplay –Davb:eavb_media …



Clock


Master
Local audio processing
audio streams
1722 packetizing & time-stamping
1722 stream containing n audio channels
Amp
1722 de-packetizing
Local audio processing
1722 time stamp extraction
local audio frequency adjustment
1722 stream containing m audio channels
1722 "null stream" containing no audio channels (for media clock transfer only) Future: Use 1722a Media Clock Stream instead
1722 time stamp extraction
Local audio processing
1722 packetizing & time-stamping
local audio frequency adjustment
Slave

1. The AVB StreamHandler is capable of analyzing the media clock rate of incoming AVB streams, based on the time stamp sequence embedded in the stream and the average packet rate. This information can be compared against the rate of a local clock domain. The result of this comparison can be used to control a local adjustable oscillator (e.g. a PLL) to recreate the media clock as a physical signal with the quality required to drive audio HW, for instance a D/A converter.
2. AVB clock domain, the default will be PTP clock. When PTP clock missing, AVB use packet timestamp to calculate with average sample rate to create media clock.
3. When network stream is NULL, AVB will use to convey media clock to local stream. Local stream will use media clock to sync local media processing to remote clock.

Configuration

Setup on BSP

Check the IP address. If the IP address is not set, follow these steps to set the IP:

1. On Master: ip addr add 192.168.1.11 dev eth0
2. On Slave: ip addr add 192.168.1.12 dev eth0

Note: This is applicable to switch setup only.

Check on services:

- GPTP needs to be up.
- Avb_streamhandler need to be running.
- For the Android* Platform, make sure the correct policies are set in Parameter firmware for master and slave.

Steps - 1

	Master	Slave
Set device mode	# setprop persist.eavb.mode m # reboot # getprop persist.eavb.mode ➔ [persist.eavb.mode]: [m]	# setprop persist.eavb.mode s # reboot # getprop persist.eavb.mode ➔ [persist.eavb.mode]: [s]
Check GPTP service*	# getprop gptp → [init.svc.gptp_a]: [running]	# getprop gptp → [init.svc.gptp_as]: [running]
Set GPTP profile*	# getprop gptp → [persist.gptp.automotive_profile]: [y]	# getprop gptp → [persist.gptp.automotive_profile]: [y]

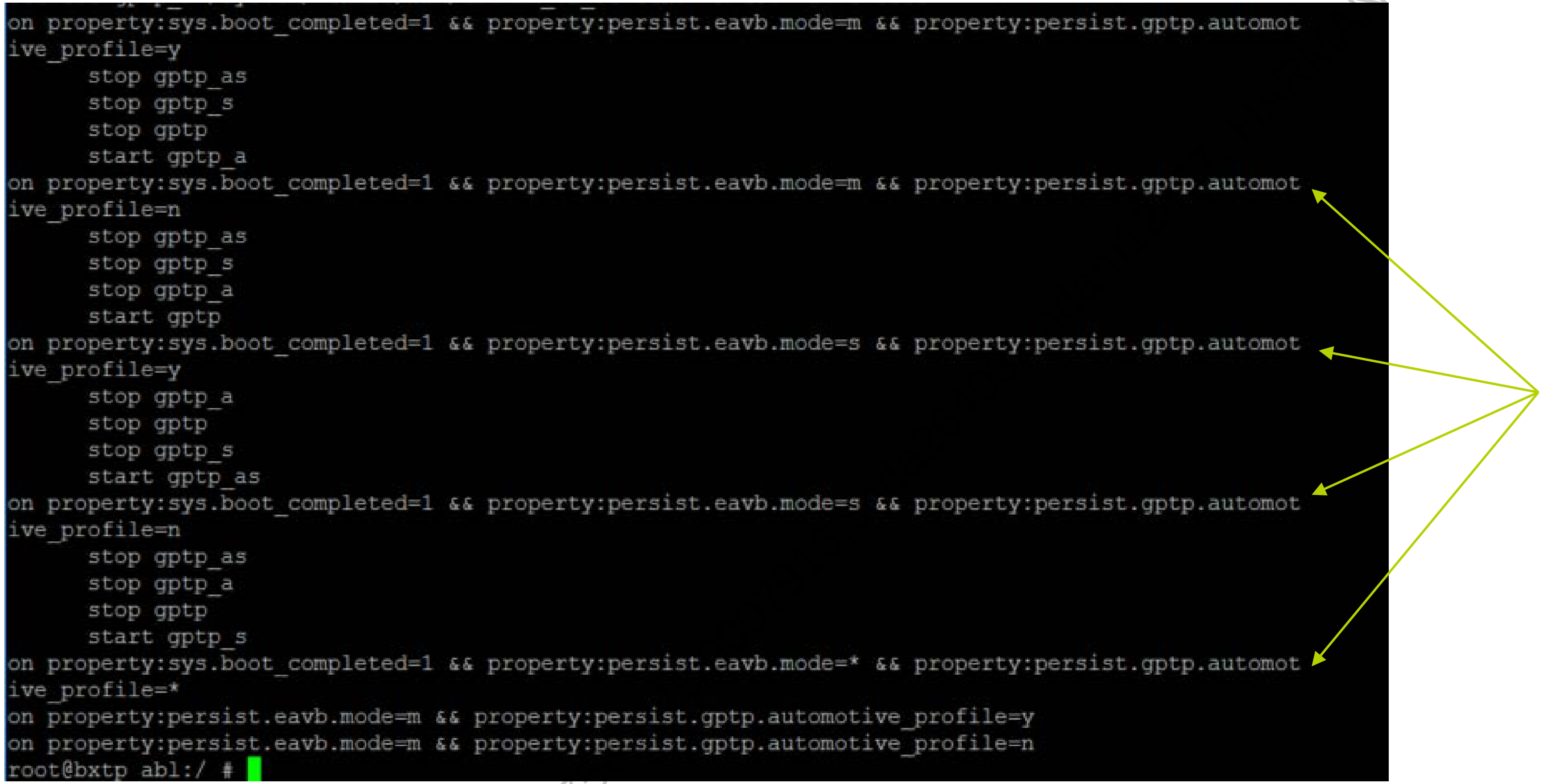
* The selection of the GPTP service and profile is highly dependent on the eAVB connection setup. Refer to Profile and Service Configuration for detailed information.

Steps - 2

	Master	Slave
Set AVB Clock Domain*	**To set RAW Clock Domain:** # setprop persist.avb.clock.raw y # getprop | grep avb → [persist.audio.audioConf]: [AudioParameterFramework-td8532-eavb-master-raw.xml] → [persist.avb.clock.raw]: [y] → [persist.avb.profile.name]: [MRB_Master_Audio_Raw] **To set RAW Clock Domain:** # setprop persist.avb.clock.raw n # getprop | grep avb → [persist.audio.audioConf]: [AudioParameterFramework-td8532-eavb-master.xml] → [persist.avb.clock.raw]: [n] → [persist.avb.profile.name]: [MRB_Master_Audio]	N/A
Set D6 avb draft	# getprop | grep d6.mode → [persist.d6.mode]: [y]	# getprop | grep d6.mode → [persist.d6.mode]: [y]

* The clock domain is the reference media clock. For Android* M, this value is automatically set when thegptp.automotive_profile is set to true.

Profile and Service Configuration



```
on property:sys.boot_completed=1 && property:persist.eavb.mode=m && property:persist.gptp.automot
ive_profile=y
        stop gptp_as
        stop gptp_s
        stop gptp
        start gptp_a
on property:sys.boot_completed=1 && property:persist.eavb.mode=m && property:persist.gptp.automot
ive_profile=n
        stop gptp_as
        stop gptp_s
        stop gptp_a
        start gptp
on property:sys.boot_completed=1 && property:persist.eavb.mode=s && property:persist.gptp.automot
ive_profile=y
        stop gptp_a
        stop gptp
        stop gptp_s
        start gptp_as
on property:sys.boot_completed=1 && property:persist.eavb.mode=s && property:persist.gptp.automot
ive_profile=n
        stop gptp_as
        stop gptp_a
        stop gptp
        start gptp_s
on property:sys.boot_completed=1 && property:persist.eavb.mode=* && property:persist.gptp.automot
ive_profile=*
on property:persist.eavb.mode=m && property:persist.gptp.automotive_profile=y
on property:persist.eavb.mode=m && property:persist.gptp.automotive_profile=n
root@bxtp_abl:/ #
```

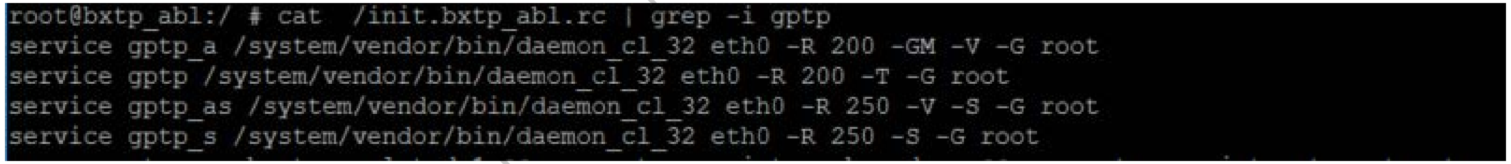
The configuration is highly dependent on the connection setup. The connection can be:

- Direct connect
- Basic AVB Switch

A combination of properties will configure the eAVB into a different mode

GPTP Configuration

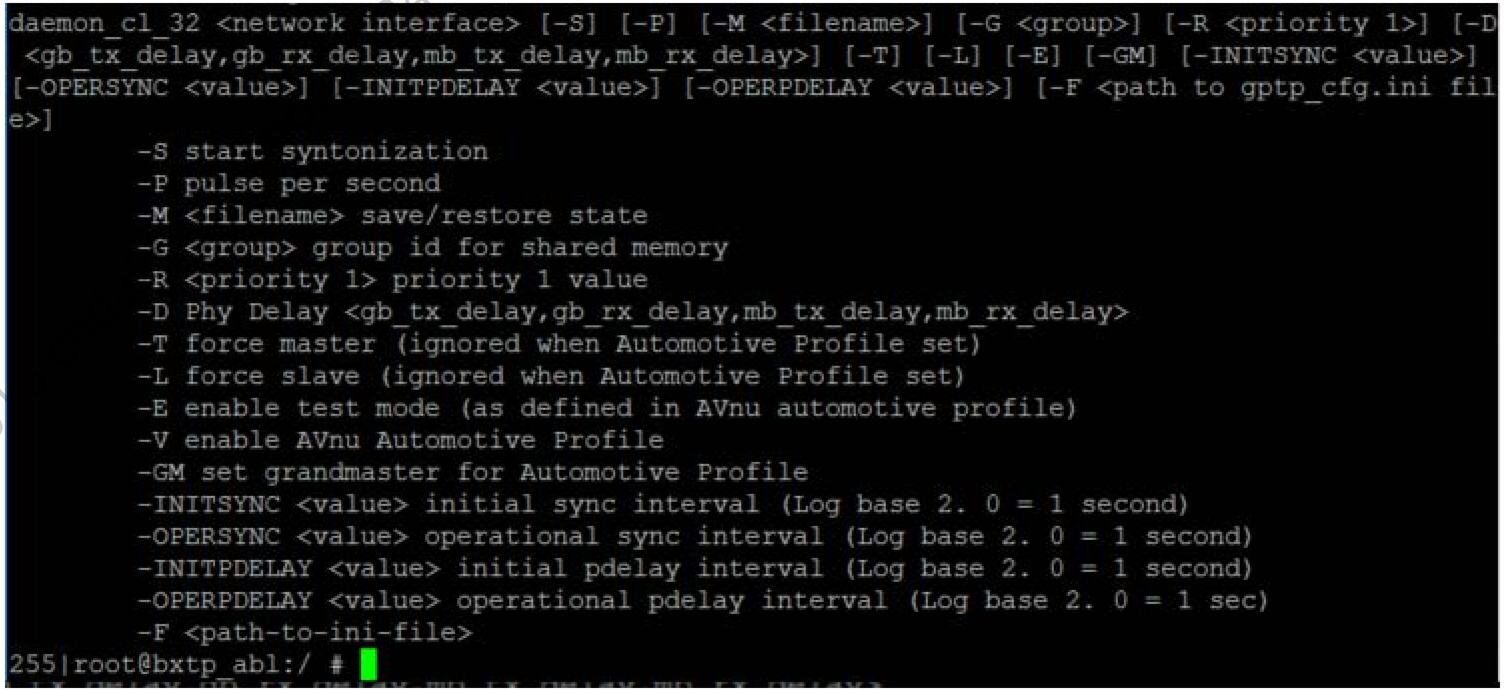
4 GPTP service profiles are predefined



```
root@bxtp_abl:/ # cat  /init.bxtp_abl.rc | grep -i gptp
service gptp_a /system/vendor/bin/daemon_cl_32 eth0 -R 200 -GM -V -G root
service gptp /system/vendor/bin/daemon_cl_32 eth0 -R 200 -T -G root
service gptp_as /system/vendor/bin/daemon_cl_32 eth0 -R 250 -V -S -G root
service gptp_s /system/vendor/bin/daemon_cl_32 eth0 -R 250 -S -G root
```

Profile Definition

- gptp_a – set as priority 1, as grandmaster for automotive profile, enable Avnu automotive profile.
- gptp - set as priority 1, force master.
- gptp_as – set as priority 1, enable Avnu automotive profile and start syntonization.
- gptp_s - set as priority 1, and start syntonization.



```
daemon_cl_32 <network interface> [-S] [-P] [-M <filename>] [-G <group>] [-R <priority 1>] [-D
 <gb_tx_delay,gb_rx_delay,mb_tx_delay,mb_rx_delay>] [-T] [-L] [-E] [-GM] [-INITSYNC <value>]
[-OPERSYNC <value>] [-INITPDELAY <value>] [-OPERPDELAY <value>] [-F <path to gptp_cfg.ini fil
e>]
        -S start syntonization
        -P pulse per second
        -M <filename> save/restore state
        -G <group> group id for shared memory
        -R <priority 1> priority 1 value
        -D Phy Delay <gb_tx_delay,gb_rx_delay,mb_tx_delay,mb_rx_delay>
        -T force master (ignored when Automotive Profile set)
        -L force slave (ignored when Automotive Profile set)
        -E enable test mode (as defined in AVnu automotive profile)
        -V enable AVnu Automotive Profile
        -GM set grandmaster for Automotive Profile
        -INITSYNC <value> initial sync interval (Log base 2. 0 = 1 second)
        -OPERSYNC <value> operational sync interval (Log base 2. 0 = 1 second)
        -INITPDELAY <value> initial pdelay interval (Log base 2. 0 = 1 second)
        -OPERPDELAY <value> operational pdelay interval (Log base 2. 0 = 1 sec)
        -F <path-to-ini-file>
255|root@bxtp_abl:/ #
```

Check Board Synchronization

- Go to /system/bin
- Run open_avb_shm_test on both boards.
  - If **asCapable** is set as True, it is working.
  - If **asCapable** is not set as True, run the following in both boards.
    - stop avbstreamhandler
    - stop gptp*
    - start gptp*
    - start avbstreamhandler
  - Port state for master 7
  - Port state for slave 9

* The gptp service depends on which configuration has been set. (gptp_a, gptp_as, gptp_s)

Customization

Customized Configuration

The configuration of the AVB StreamHandler is done through a configuration plugin shared object library used by the AVB StreamHandler to preconfigure the AVB network and local streams during startup.

A reference implementation of the plugin containing example targets and profiles is provided with the AVB StreamHandler (pluginias-media_transportavb_configuration_reference.so). This reference plugin is loaded by the avb_streamhandler_app during startup if no other plugin is specified.

Configuration Plugin

The configuration plugin serves three purposes:

1. Determine configuration options.

   - The configuration plugin can access all parameters of the AVB StreamHandler invocation on the right hand side of the keyword setup. It also could use any other mechanism available to determine configuration options. This includes environment variables, file access, and so on. It is up to the plugin implementer to determine such mechanisms.

   - This task should be done in the context of the passArguments() method, which must be implemented by the configuration object.

Configuration Plugin

2. Set entries in the configuration registry.

- The plugin can set textual or numerical entries on a key->value basis in a registry database provided by the AVB StreamHandler. The key names are provided in the public header file IasAvbRegistryKeys.hpp. Customers can also create their own keys to pass configuration options through to a clock library implementation.

- This task should be done in the context of the passArguments() method that must be implemented by the configuration object. After passArguments() returns, no more entries can be added to the registry.

Configuration Plugin

3. Use the IasAvbStreamHandlerInterface API to set up the operating environment for the AVB StreamHandler.

    - For example, creating AVB and local streams, connecting them, and defining the operating parameters for clock recovery. Note that the clock recovery settings can only be controlled by the configuration plugin, not by the API.

Configuration Plugin

- In the configuration plugin, all definitions regarding the streams, their parameters, and stream-related registry entries are listed as **Profiles**.
- All definitions related to the underlying HW system (for example, PCIe* device ID, bus address, and network interface name) are defined as **Targets**.
- The following steps outline the modifications for a customized library. All the folders in the section below are located under: `$vendor/intel/ias/media_transport/avb_streamhandler`

Customization Steps - 1

Create a new makefile to build a customer-specific library.

1. `cd android/`
2. `cp AvbConfigurationReference.mk AvbConfigurationcustomer.mk`
3. Edit the new AvbConfigurationCustomer.mk file to point to the new source library that will be created:

```
LOCAL_MODULE := pluginias-media_transport-avb_configuration_customer
LOCAL_SRC_FILES :=
../private/src/avb_configuration_customer/IasAvbConfigcustomer.cpp
```

Customization Steps - 2

Include the new makefile in the make process.

Include the new customer configuration file in the Android.mk file:

```
include $(LOCAL_PATH)/AvbConfigurationcustomer.mk
```

Customization Steps - 3

Create the custom configuration file.

Create a customer-specific configuration library source:

```
1. cd private/src
2. mkdir avb_configuration_customer
3. cp ./avb_configuration_reference/IasAvbConfigExample.cpp
./avb_configuration_customer/IasAvbConfigcustomer.cpp
4. cd avb_configuration_customer
```

Customization Steps - 4

Edit IasAvbConfigcustomer.cpp.

Remember that the IasAVBConfigExample.cpp file is susceptible to change from release to release.

We recommend modifying the file that is part of the release and locating the code to be added or modified. The sections in bold are likely to change depending on the platform and the need for modification based on the HW.

Customization Steps - 5

Assign the PtPClockdomain or raw clock domain to the master clock.

```
static const Ias::UInt32 cRefClockId = 0x80864711u;
static const Ias::UInt32 cMasterClockId =
cIasAvbRawClockDomainId;
```

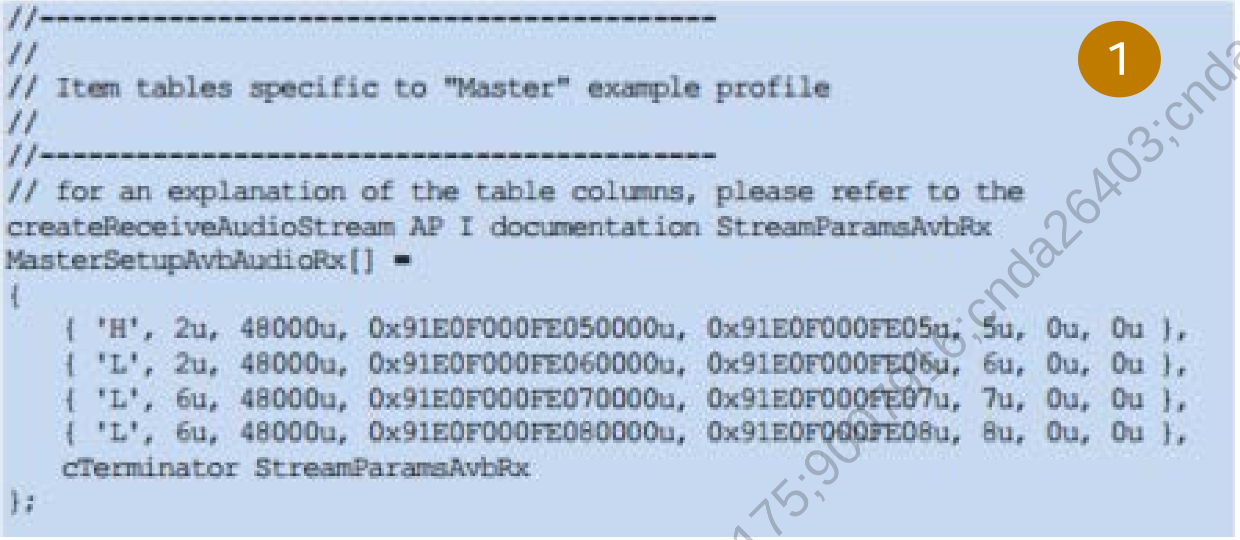
or

```
static const Ias::UInt32 cMasterClockId =
cIasAvbPtpClockDomainId;
```

Customization Steps - 6

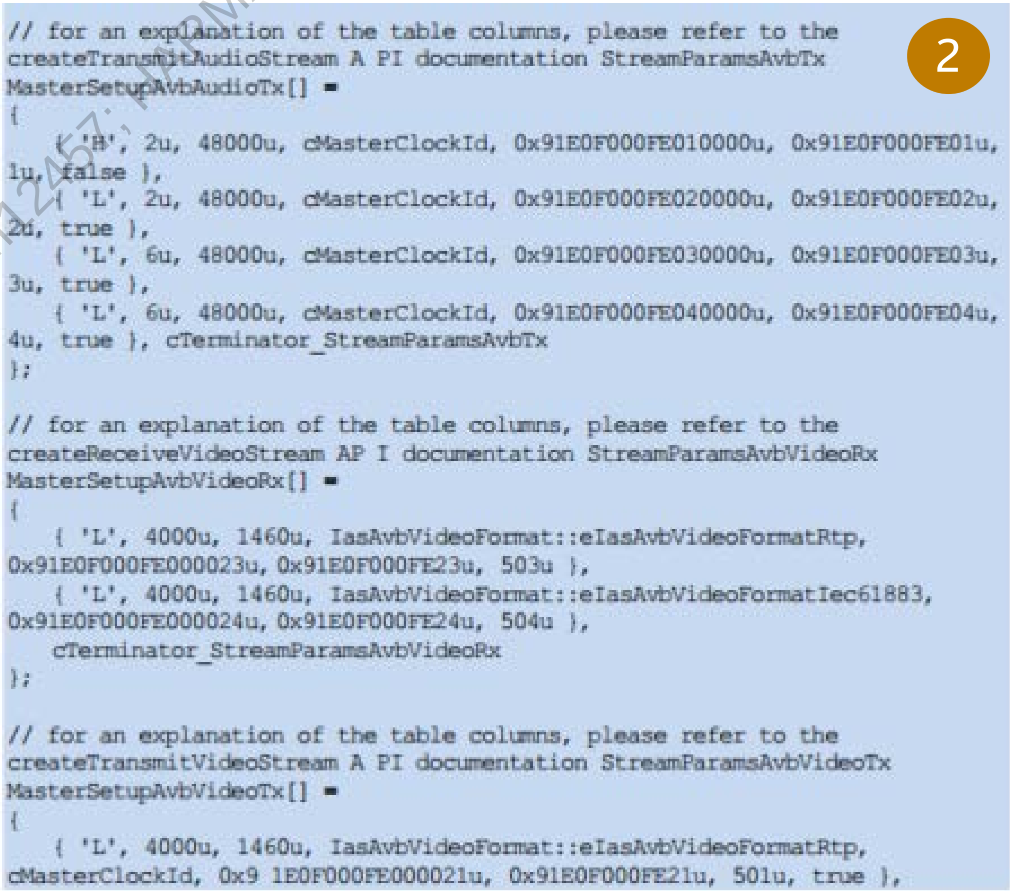
Add the customer-specific configuration for their streams to the new configuration file.

Add the profiles here:



1

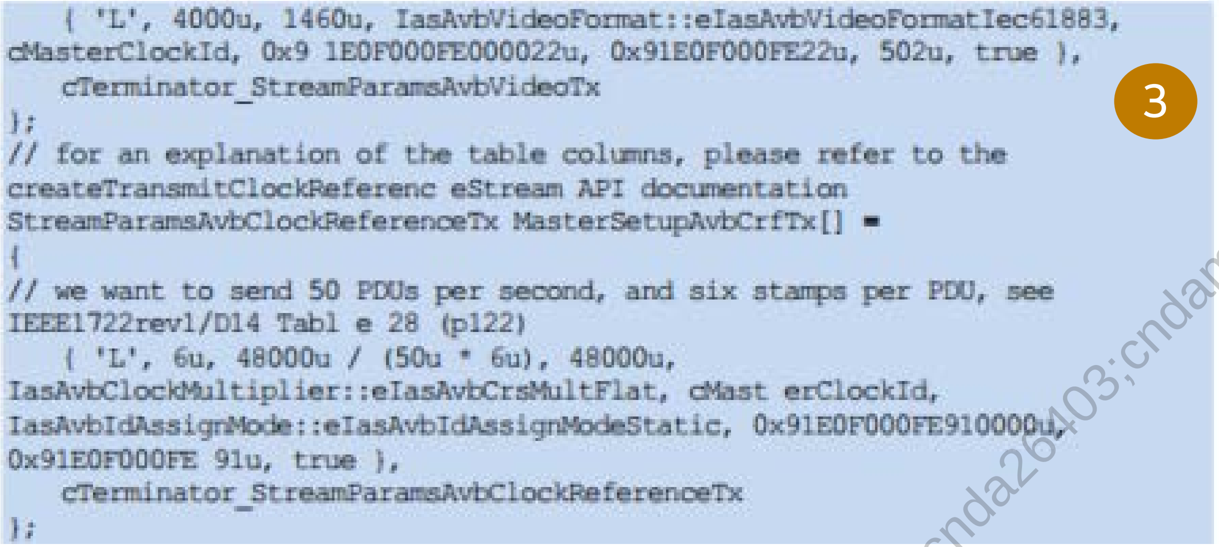
```
//-------------------------------------------
//
// Item tables specific to "Master" example profile
//
//-------------------------------------------
// for an explanation of the table columns, please refer to the
createReceiveAudioStream AP I documentation StreamParamsAvbRx
MasterSetupAvbAudioRx[] =
{
   { 'H', 2u, 48000u, 0x91E0F000FE050000u, 0x91E0F000FE05u, 5u, 0u, 0u },
   { 'L', 2u, 48000u, 0x91E0F000FE060000u, 0x91E0F000FE06u, 6u, 0u, 0u },
   { 'L', 6u, 48000u, 0x91E0F000FE070000u, 0x91E0F000FE07u, 7u, 0u, 0u },
   { 'L', 6u, 48000u, 0x91E0F000FE080000u, 0x91E0F000FE08u, 8u, 0u, 0u },
   cTerminator StreamParamsAvbRx
};
```



2

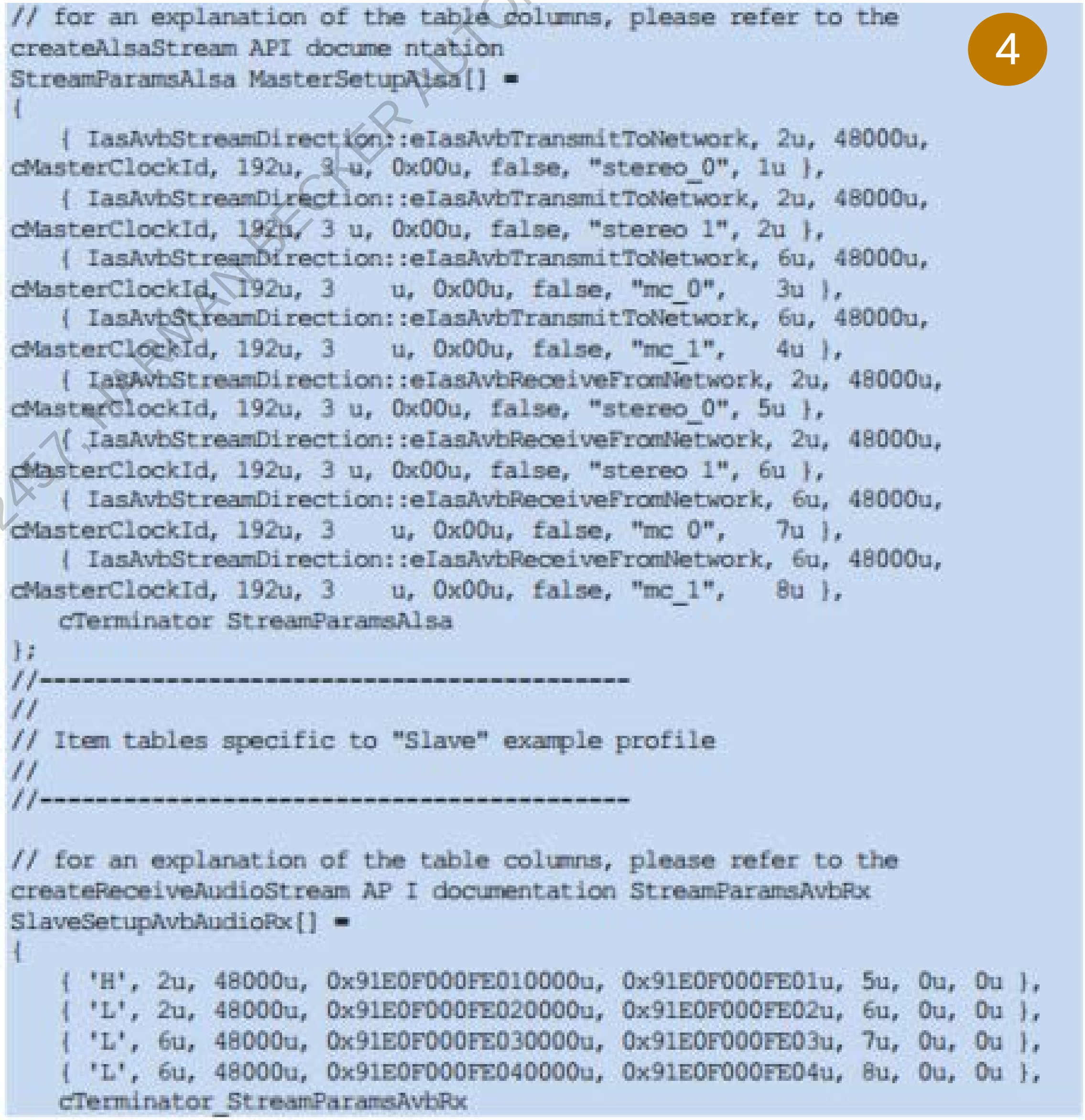
```
// for an explanation of the table columns, please refer to the
createTransmitAudioStream A PI documentation StreamParamsAvbTx
MasterSetupAvbAudioTx[] =
{
   { 'H', 2u, 48000u, cMasterClockId, 0x91E0F000FE010000u, 0x91E0F000FE01u,
1u, false },
   { 'L', 2u, 48000u, cMasterClockId, 0x91E0F000FE020000u, 0x91E0F000FE02u,
2u, true },
   { 'L', 6u, 48000u, cMasterClockId, 0x91E0F000FE030000u, 0x91E0F000FE03u,
3u, true },
   { 'L', 6u, 48000u, cMasterClockId, 0x91E0F000FE040000u, 0x91E0F000FE04u,
4u, true }, cTerminator_StreamParamsAvbTx
};

// for an explanation of the table columns, please refer to the
createReceiveVideoStream AP I documentation StreamParamsAvbVideoRx
MasterSetupAvbVideoRx[] =
{
   { 'L', 4000u, 1460u, IasAvbVideoFormat::eIasAvbVideoFormatRtp,
0x91E0F000FE000023u, 0x91E0F000FE23u, 503u },
   { 'L', 4000u, 1460u, IasAvbVideoFormat::eIasAvbVideoFormatIec61883,
0x91E0F000FE000024u, 0x91E0F000FE24u, 504u },
   cTerminator_StreamParamsAvbVideoRx
};

// for an explanation of the table columns, please refer to the
createTransmitVideoStream A PI documentation StreamParamsAvbVideoTx
MasterSetupAvbVideoTx[] =
{
   { 'L', 4000u, 1460u, IasAvbVideoFormat::eIasAvbVideoFormatRtp,
cMasterClockId, 0x9 1E0F000FE000021u, 0x91E0F000FE21u, 501u, true },
```

Customization Steps - 7



3

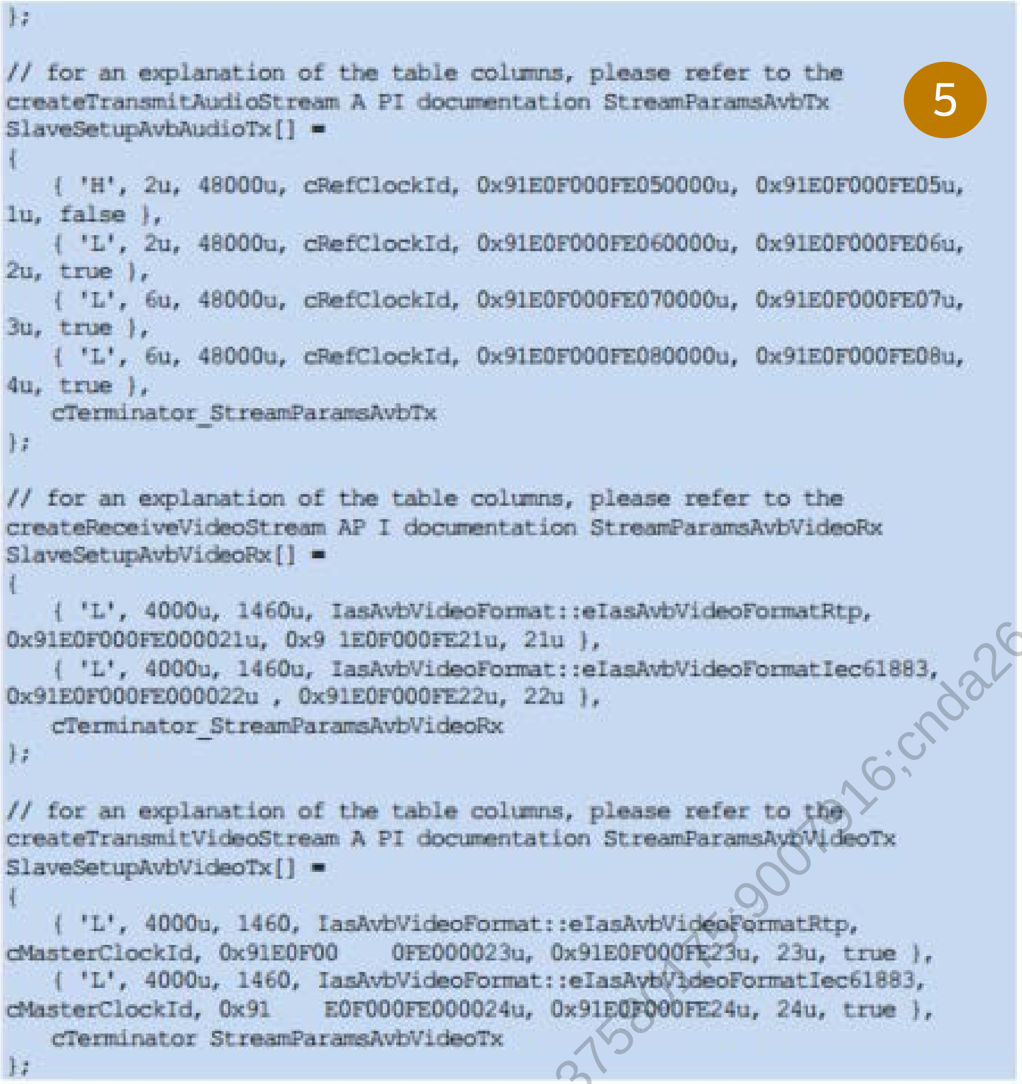
```
    { 'L', 4000u, 1460u, IasAvbVideoFormat::eIasAvbVideoFormatIec61883,
cMasterClockId, 0x9 1E0F000FE000022u, 0x91E0F000FE22u, 502u, true },
    cTerminator_StreamParamsAvbVideoTx
};
// for an explanation of the table columns, please refer to the
createTransmitClockReferenc eStream API documentation
StreamParamsAvbClockReferenceTx MasterSetupAvbCrfTx[] =
{
// we want to send 50 PDUs per second, and six stamps per PDU, see
IEEE1722rev1/D14 Tabl e 28 (p122)
    { 'L', 6u, 48000u / (50u * 6u), 48000u,
IasAvbClockMultiplier::eIasAvbCrsMultFlat, cMast erClockId,
IasAvbIdAssignMode::eIasAvbIdAssignModeStatic, 0x91E0F000FE910000u,
0x91E0F000FE 91u, true },
    cTerminator_StreamParamsAvbClockReferenceTx
};
```



4

```
// for an explanation of the table columns, please refer to the
createAlsaStream API docume ntation
StreamParamsAlsa MasterSetupAlsa[] =
{
    { IasAvbStreamDirection::eIasAvbTransmitToNetwork, 2u, 48000u,
cMasterClockId, 192u, 3 u, 0x00u, false, "stereo_0", 1u },
    { IasAvbStreamDirection::eIasAvbTransmitToNetwork, 2u, 48000u,
cMasterClockId, 192u, 3 u, 0x00u, false, "stereo 1", 2u },
    { IasAvbStreamDirection::eIasAvbTransmitToNetwork, 6u, 48000u,
cMasterClockId, 192u, 3    u, 0x00u, false, "mc_0",    3u },
    { IasAvbStreamDirection::eIasAvbTransmitToNetwork, 6u, 48000u,
cMasterClockId, 192u, 3    u, 0x00u, false, "mc_1",    4u },
    { IasAvbStreamDirection::eIasAvbReceiveFromNetwork, 2u, 48000u,
cMasterClockId, 192u, 3 u, 0x00u, false, "stereo_0", 5u },
    { IasAvbStreamDirection::eIasAvbReceiveFromNetwork, 2u, 48000u,
cMasterClockId, 192u, 3 u, 0x00u, false, "stereo 1", 6u },
    { IasAvbStreamDirection::eIasAvbReceiveFromNetwork, 6u, 48000u,
cMasterClockId, 192u, 3    u, 0x00u, false, "mc 0",    7u },
    { IasAvbStreamDirection::eIasAvbReceiveFromNetwork, 6u, 48000u,
cMasterClockId, 192u, 3    u, 0x00u, false, "mc_1",    8u },
    cTerminator StreamParamsAlsa
};
//-------------------------------------------
//
// Item tables specific to "Slave" example profile
//
//-------------------------------------------

// for an explanation of the table columns, please refer to the
createReceiveAudioStream AP I documentation StreamParamsAvbRx
SlaveSetupAvbAudioRx[] =
{
    { 'H', 2u, 48000u, 0x91E0F000FE010000u, 0x91E0F000FE01u, 5u, 0u, 0u },
    { 'L', 2u, 48000u, 0x91E0F000FE020000u, 0x91E0F000FE02u, 6u, 0u, 0u },
    { 'L', 6u, 48000u, 0x91E0F000FE030000u, 0x91E0F000FE03u, 7u, 0u, 0u },
    { 'L', 6u, 48000u, 0x91E0F000FE040000u, 0x91E0F000FE04u, 8u, 0u, 0u },
    cTerminator_StreamParamsAvbRx
```

Customization Steps - 8



5

```
};

// for an explanation of the table columns, please refer to the
createTransmitAudioStream A PI documentation StreamParamsAvbTx
SlaveSetupAvbAudioTx[] =
{
   { 'H', 2u, 48000u, cRefClockId, 0x91E0F000FE050000u, 0x91E0F000FE05u,
1u, false },
   { 'L', 2u, 48000u, cRefClockId, 0x91E0F000FE060000u, 0x91E0F000FE06u,
2u, true },
   { 'L', 6u, 48000u, cRefClockId, 0x91E0F000FE070000u, 0x91E0F000FE07u,
3u, true },
   { 'L', 6u, 48000u, cRefClockId, 0x91E0F000FE080000u, 0x91E0F000FE08u,
4u, true },
   cTerminator_StreamParamsAvbTx
};

// for an explanation of the table columns, please refer to the
createReceiveVideoStream AP I documentation StreamParamsAvbVideoRx
SlaveSetupAvbVideoRx[] =
{
   { 'L', 4000u, 1460u, IasAvbVideoFormat::eIasAvbVideoFormatRtp,
0x91E0F000FE000021u, 0x9 1E0F000FE21u, 21u },
   { 'L', 4000u, 1460u, IasAvbVideoFormat::eIasAvbVideoFormatIec61883,
0x91E0F000FE000022u , 0x91E0F000FE22u, 22u },
   cTerminator_StreamParamsAvbVideoRx
};

// for an explanation of the table columns, please refer to the
createTransmitVideoStream A PI documentation StreamParamsAvbVideoTx
SlaveSetupAvbVideoTx[] =
{
   { 'L', 4000u, 1460, IasAvbVideoFormat::eIasAvbVideoFormatRtp,
cMasterClockId, 0x91E0F00    0FE000023u, 0x91E0F000FE23u, 23u, true },
   { 'L', 4000u, 1460, IasAvbVideoFormat::eIasAvbVideoFormatIec61883,
cMasterClockId, 0x91    E0F000FE000024u, 0x91E0F000FE24u, 24u, true },
   cTerminator StreamParamsAvbVideoTx
};
```

Note: If the system requires 1722a-D6 mode, this can be enabled in the registry setup. Table 3 describes the related keys. It is common practice to adjust these parameters to the parameters of the SR class needed in a project:

Key	Description	Default (High)	Default (Low)
tspec.vlanid	VLAN ID	2	3
tspec.vlanprio	VLAN priority (PCP)	3	2
tspec.presentation.time.offset	Maximum Transit Time	1875000ns	9750000ns
tx.maxbandwidth	Maximum Bandwidth	70000kbits/sec	70000kbits/sec

Table 3: Key Settings for 1722a-D6 Mode

Customization Steps - 9

6

```
RegistryEntries ExampleSetupRegistry[] =
{
    // define the "low" class according to "class C"
    { "tspec.interval.low", true, 1333000u, NULL }, // class C measurement
interval (1.333ms)
    { "tspec.vlanid.low", true, 2u, NULL },  // VLAN ID
    { "tspec.vlanprio.low", true, 3u, NULL },   // VLAN priority (PCP)
    { "tspec.presentation.time.offset.low", true, 15000000u, NULL }, //
Setting to 15ms per GM requirements

    // set bandwidth limit for all active class C streams to 50Mbit/s
    { "tx.maxbandwidth.low", true, 50000u, NULL },
    { "transmit.window.width", true, 6000000u, NULL },
    { "transmit.window.pitch", true, 4000000u, NULL },

    // set RX engine to wait up to 50ms for new packets to avoid premature
timeout for video-only scenarios
    { "receive.idlewait", true, 50000000u, NULL },

    // set AVTP audio format to 1722a-D6 (affects ALL RX and TX audio
streams!)
    {"compatibility.audio", false, 0u, "d6_1722a" },
    {"local.alsa.baseperiod", true, 384u, NULL },
    {"local.alsa.ringbuffer", true, 12288u, NULL },
    cTerminator_RegistryEntries
};
```

Add the target parameters:

7

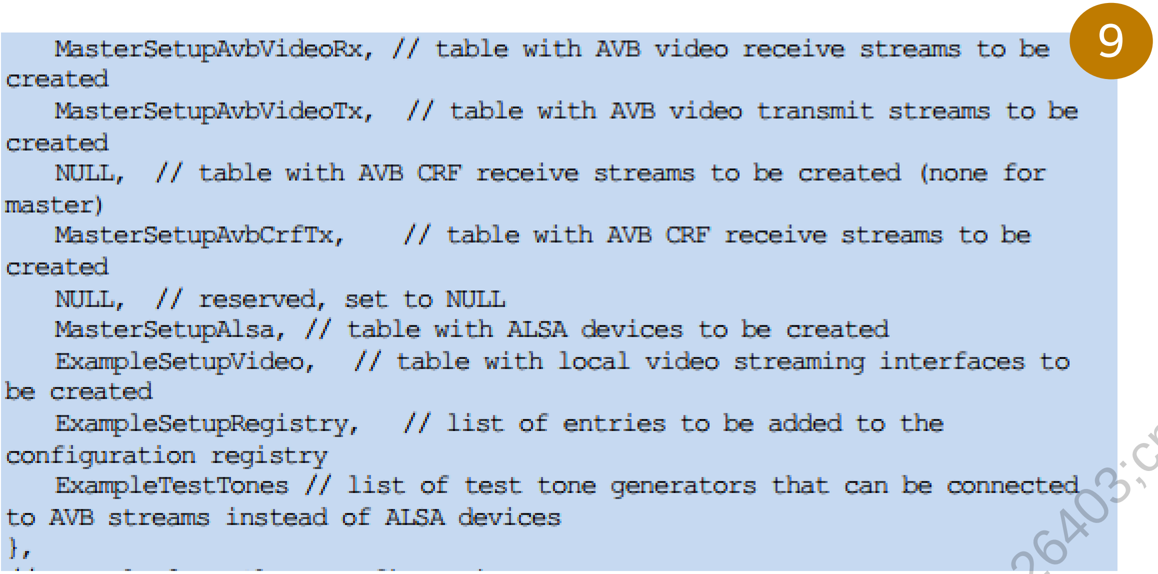
```
TargetParams Targets[] =
{
    { "GrMrb", 0x1533u, 2u, "eth0", NULL }, // example entry for Gordon
Ridge MRB
    {<//enter customer hardware entries here>},
};
```

Add the profile parameters:

8

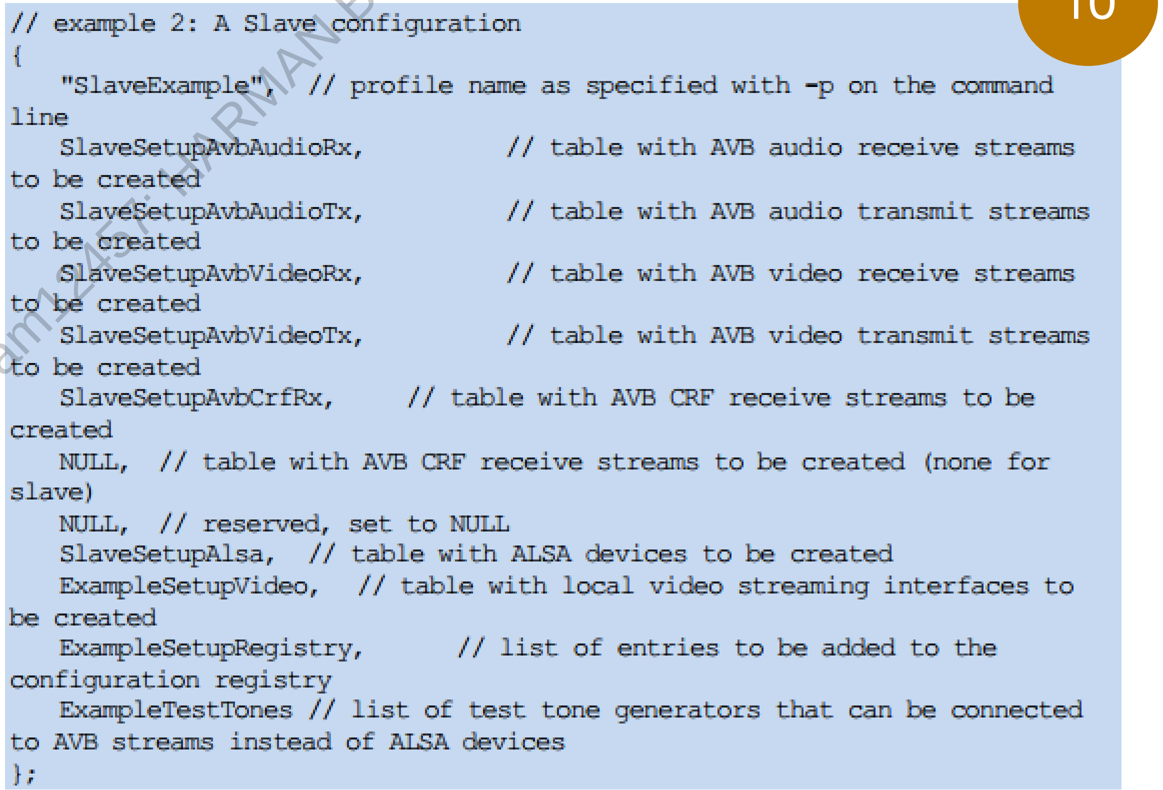
```
{
    "MasterExample", // profile name as specified with -p on the command
line
    MasterSetupAvbAudioRx, // table with AVB audio receive streams to be
created
    MasterSetupAvbAudioTx, // table with AVB audio transmit streams to be
created
```

Customization Steps - 10



9

```
    MasterSetupAvbVideoRx, // table with AVB video receive streams to be
created
    MasterSetupAvbVideoTx,  // table with AVB video transmit streams to be
created
    NULL,  // table with AVB CRF receive streams to be created (none for
master)
    MasterSetupAvbCrfTx,    // table with AVB CRF receive streams to be
created
    NULL,  // reserved, set to NULL
    MasterSetupAlsa, // table with ALSA devices to be created
    ExampleSetupVideo,   // table with local video streaming interfaces to
be created
    ExampleSetupRegistry,   // list of entries to be added to the
configuration registry
    ExampleTestTones // list of test tone generators that can be connected
to AVB streams instead of ALSA devices
},
```



10

```
// example 2: A Slave configuration
{
    "SlaveExample",  // profile name as specified with -p on the command
line
    SlaveSetupAvbAudioRx,          // table with AVB audio receive streams
to be created
    SlaveSetupAvbAudioTx,          // table with AVB audio transmit streams
to be created
    SlaveSetupAvbVideoRx,          // table with AVB video receive streams
to be created
    SlaveSetupAvbVideoTx,          // table with AVB video transmit streams
to be created
    SlaveSetupAvbCrfRx,     // table with AVB CRF receive streams to be
created
    NULL,  // table with AVB CRF receive streams to be created (none for
slave)
    NULL,  // reserved, set to NULL
    SlaveSetupAlsa,  // table with ALSA devices to be created
    ExampleSetupVideo,  // table with local video streaming interfaces to
be created
    ExampleSetupRegistry,      // list of entries to be added to the
configuration registry
    ExampleTestTones // list of test tone generators that can be connected
to AVB streams instead of ALSA devices
};
```

Log / Trace Message

Log Level

There are six different log levels:

0: off   1: fatal   2: error   3: warning   4: info   5: debug   6: verbose

The default log level of the AVB StreamHandler is 'warning' (3). The log level for all contexts can be increased from the default level 'warning' by adding the desired numbers of v in the command line (somewhere before the word 'setup'). For 'info' (4) use -v, for 'debug' (5) use -vv and for 'verbose' (6) use -vvv. The same effect is achieved by setting the level in the DLT viewer. It is not possible to decrease the log level below the default level 'warning' by using -v option.

Important Messages

- **Reset due to launch time lag**

  If the package time is in the past or max transit time can not be guaranteed, the stream gets reset and following error DLT messages appear (numbers will differ) This leads always to hearable audio drops.

- **TX worker thread slept too long!**

  The worker thread was not activated by the Linux* scheduler in time. There is some safety margin calculated by window size(5ms) - window pitch(3ms) = 2ms. If the safety margin has completely run out, underruns may occur. Values are adjustable depending on system setup.

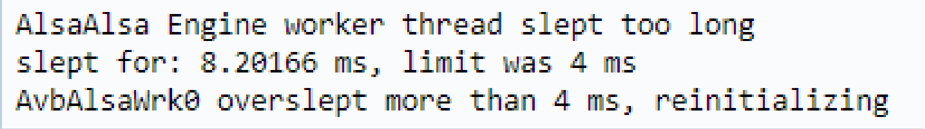
- **<numEvents> more oversleep events occurred since the last message**

  If those errors happen often after each other, they will be collected and the count given.

Important Messages

- **ALSA engine worker thread slept too long**

Example:



```
AlsaAlsa Engine worker thread slept too long
slept for: 8.20166 ms, limit was 4 ms
AvbAlsaWrk0 overslept more than 4 ms, reinitializing
```

  - The worker thread was not activated by the Linux* scheduler in time. The thread is expected to run at period time interval, which is 4 ms in the above case.
  - If activation time was more than a period time late, it will restart the thread, which may cause underruns (playback) or overruns (capture).
  - You may change maximum allowable over sleep time with '-k alsa.clock.threshold.reset=<ns>' key.
  - For example, if the key value was 8000000 ns (=8 ms), the thread would recover the delayed periods up to 8 ms when it wakes up.
  - It may momentarily increase CPU load in order to process pending periods in a rush. The key value should not exceed the max buffer time of the ALSA stream. That means for a stream with 48 kHz freq, 192 period size and 3 periods, the maximum key value should be 12000000 ns (=12 ms).

Tools

Debugging Tools

- DLT (Yocto*)
- MPTA (https://wiki.ith.intel.com/pages/viewpage.action?pageId=196316291)
- Logcat*
- Wireshark*
- Wsd-tool – Wireshark* log extraction

Reference

Links

DLT:

http://asd.ka.intel.com/kc/doc/live/KC3.0/doc/media_transport/avb_streamhandler/md_51_Analysis.html#dlt

MPTA:

https://wiki.ith.intel.com/pages/viewpage.action?pageId=196316291575560-

Application Note:

Gordon Peak for Android* Ethernet Audio Video Bridge (eAVB) Profile Configuration: 575560-gp-for-android-eavb-profile-config-app-note-rev0-5.pdf

Backup