

# **Intel® Smart Sound Technology Audio DSP Non-HD Audio ACPI High Level Design**

**Architecture Guide/Overview**

---

***February 2016***

***Revision 0.7***



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.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. **No computer system can be absolutely secure.** Check with your system manufacturer or retailer or learn more at [intel.com](http://intel.com).

Intel technologies may require enabled hardware, specific software, or services activation. Check with your system manufacturer or retailer.

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.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

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

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

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

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

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

Copyright © 2015-2016, Intel Corporation. All rights reserved.



# Contents

1	Introduction .....	6
1.1	Purpose of this Document .....	6
1.2	Document Scope.....	6
1.3	Terminology .....	6
1.4	Reference Documents.....	7
2	Non-HD Audio Endpoint Description Table.....	8
2.1	Instance ID versus Virtual Bus ID .....	10
2.2	NHLT Element Descriptions .....	10
2.3	Device Capabilities Structure .....	13
2.4	Multi-channel Transmission .....	14
2.5	Microphone Array Device Capabilities Structure .....	15
2.6	Microphone Array Coordinates .....	17
2.6.1	Front (Main) Microphone Array Coordinates .....	17
2.6.2	Rear (Additional) Microphone Array Coordinates .....	18
2.7	Render Device with Loopback .....	19
2.8	Multi-mode Stream Types .....	19
2.9	Multi-mode Endpoint .....	19
2.10	Multi-mode Capture Microphone Array .....	20
3	Examples.....	21
3.1	DMIC Connected via PDM Interface .....	21
3.2	DMIC Connected via I2S Codec.....	22
3.3	BT Sideband Device Connected via I2S Port .....	24
3.4	SSP (TDM Mode) Device ACPI Configuration .....	26
3.5	Sample NHLT Example .....	30
3.5.1	Platform Connectivity .....	30
3.5.2	NHLT Table .....	31

## Figures

Figure 2-1. Table Layout.....	9
Figure 2-2 Coordinate System of Front (Main) Laptop Microphone Array .....	17
Figure 2-3 Coordinate System of Front (Main) Tablet Microphone Array .....	17
Figure 2-4 Coordinate System Of Rear (Additional) Laptop Microphone Array .....	18
Figure 2-5 Coordinate System of Rear (Additional) Tablet Microphone Array .....	18
Figure 3-1. SKL Reference Platform Connectivity .....	31

## Tables

Table 2-1. NHLT Descriptor .....	9
Table 2-2. WaveFormatExtensible.....	11



Table 2-3. ArrayType Description .....	15
Table 2-4. Microphone Type Description .....	16
Table 2-5. Panel (Location of the Microphone) .....	16
Table 3-1. ENDPOINT_DESCRIPTOR [0] .....	21
Table 3-2. MIC_ARRAY_DEVICE_SPECIFIC_CONFIG (SPECIFIC_CONFIG) .....	21
Table 3-3. FORMAT_CONFIG .....	22
Table 3-4. ENDPOINT_DESCRIPTOR [0] .....	22
Table 3-5. SPECIFIC_CONFIG .....	23
Table 3-6. FORMAT_CONFIG .....	23
Table 3-7. ENDPOINT_DESCRIPTOR[1] .....	23
Table 3-8. VENDOR_MIC_ARRAY_DEVICE_SPECIFIC_CONFIG (SPECIFIC_CONFIG) ...	23
Table 3-9. ENDPOINT_DESCRIPTOR [0] (Render) .....	24
Table 3-10. SPECIFIC_CONFIG .....	24
Table 3-11. FORMAT_CONFIG .....	25
Table 3-12. ENDPOINT_CONFIGURATION [1] (Capture) .....	25
Table 3-13. SPECIFIC_CONFIG .....	25
Table 3-14. FORMAT_CONFIG .....	26
Table 3-15. ENDPOINT_CONFIGURATION[0] (Headset) .....	26
Table 3-16. DEVICE_SPECIFIC_CONFIG (Headset) .....	27
Table 3-17. FORMAT_CONFIG (Headset) .....	27
Table 3-18. ENDPOINT_CONFIGURATION [1] (Speakers) .....	27
Table 3-19. DEVICE_SPECIFIC_CONFIG (Speakers) .....	28
Table 3-20. FORMAT_CONFIG (Speakers) .....	28
Table 3-21. ENDPOINT_CONFIGURATION [2] (DMIC0,1) .....	28
Table 3-22. DEVICE_SPECIFIC_CONFIG (DMIC0,1) .....	29
Table 3-23. FORMAT_CONFIG (DMIC0,1) .....	29
Table 3-24. ENDPOINT_CONFIGURATION [3] (DMIC2,3) .....	29
Table 3-25. DEVICE_SPECIFIC_CONFIG (DMIC2,3) .....	30
Table 3-26. FORMAT_CONFIG (DMIC2,3) .....	30
Table 3-27. NHLT Table .....	31
Table 3-28. ENDPOINT_DESCRIPTOR [0] (DMICs) .....	32
Table 3-29. SPECIFIC_CONFIG .....	32
Table 3-30. FORMATS_CONFIG .....	32
Table 3-31. ENDPOINT_DESCRIPTOR [1] (Speaker Amplifiers - Playback) .....	33
Table 3-32. DEVICE_SPECIFIC_CONFIG (Speaker Amplifiers - Playback) .....	33
Table 3-33. FORMATS_CONFIG (Speaker Amplifiers - Playback) .....	33
Table 3-34. ENDPOINT_DESCRIPTOR [2] (Headset - Playback) .....	34
Table 3-35. SPECIFIC_CONFIG (Headset - Playback) .....	34
Table 3-36. FORMATS_CONFIG (Headset - Playback) .....	34
Table 3-37. ENDPOINT_DESCRIPTOR [3] (Headset - Capture) .....	34
Table 3-38. SPECIFIC_CONFIG (Headset - Capture) .....	35
Table 3-39. FORMATS_CONFIG (Headset - Capture) .....	35



## Revision History

---

Document Number	Revision Number	Description	Revision Date
561555	0.6	Initial release.	October 2015
	0.7	Clarifications added	December 2015
NA	0.7	Re-release as public document	February 2016

§ §



# 1 Introduction

---

## 1.1 Purpose of this Document

This document describes methods used in Intel® Smart Sound Technology Audio DSP products for non-HD Audio endpoints configuration performed by FW via ACPI. This configuration is applicable to both Windows\* and Linux\* based operating systems used in Skylake products. The goal is to provide information for BIOS, FW and SW engineers to implement an NHLT audio configuration.

## 1.2 Document Scope

The Intel® Smart Sound Technology Audio DSP Non-HD Audio ACPI HLD explains the structure of ACPI tables used to configure non-HD Audio endpoints connected to Audio DSP. The endpoints are connected to Audio DSP subsystem via either I2S/TDM (SSP) or PDM (DMIC). The document also describes suggested configuration method for Non-HD Audio Codecs.

## 1.3 Terminology

Term	Description
A2DP	Advance Audio Distribution Profile
ACPI	Advanced Configuration and Programming Interface version 5.0
ALSA	Advanced Linux* Sound Architecture
ASoC	ALSA System on Chip layer
Blob	Binary information which contain configuration data and other properties for each HW end point of the SKL SoC. The HW endpoints connect to specific hardware. E.g. an entry that contains DMIC coefficients for the DMIC end point.
BT/Bluetooth	Bluetooth is a local connectivity wireless protocol. Bluetooth supports the transport of voice signals (i.e. wireless headsets). Bluetooth basebands typically have a PCM audio interface for the voice data and a UART or USB for control, data, and compressed audio (using Sub-Band Coding)
DFW	Dynamic Firmware also called topology binary. This can be seen as a collection of pipelines connecting the SOC with the endpoints through the ADSP for various use cases. E.g. it has pipeline for connecting playback path to SSP0 port which in turn connects to ADI speaker. It'll also have a playback and record path to SSP1 which connects to Nuvoton codec. The pipeline connection & selection is done through the exposed mixer controls.
DSP	The firmware which is loaded to the DSP IP. This contains all the algorithms and other modules (like wake on voice / speaker protection modules. Copier module for passing data, mixers and switches to route data). For our discussion below we can assume the DSP firmware to be a constant.



Term	Description
HDMI	High Definition Multimedia Interface
LPE	Low Power Engine (Audio)
NHLT	Non HD audio Link Table. This is an ACPI table containing pointers to Blobs which are read by Kernel using the _DSM ACPI method.
Sx	Wildcard for S3/S4/S5 – Platform specific ACPI Power Management states (described in ACPI spec)
Sinks	Audio outputs / speakers
Sources	Audio inputs / microphone
HW CODEC	Hardware component which supports audio sinks and sources.
PCM	Pulse Code Modulation. Standard technique of representing an audio stream using x bits sampled uniformly y times a second. Each sample captures the amplitude of the signal at that point in time. PCM samples are sent over serial buses between processors and audio codecs
PnP	Plug and Play
SAS	Software Architecture Specification
SSP	Synchronous Serial Port – the I2S/TDM port
SST	Smart Sound Technology
VoIP	Voice over Internet Protocol

## 1.4 Reference Documents

Document	Document No./Location
Microphone array support in Windows – describes geometries and microphone type definitions	<a href="https://msdn.microsoft.com/en-us/library/windows/hardware/dn613960(v=vs.85).aspx">https://msdn.microsoft.com/en-us/library/windows/hardware/dn613960(v=vs.85).aspx</a>
Information on ACPI header	<a href="http://lxr.free-electrons.com/source/include/acpi/actbl.h#L101">http://lxr.free-electrons.com/source/include/acpi/actbl.h#L101</a>





## 2 *Non-HD Audio Endpoint Description Table*

---

Any Non-HD Audio endpoint that the platform needs to use must have a corresponding NHLT entry. If a particular non-HD Audio endpoint has been disabled, then that device or link will not be present in the ACPI configuration table.

The non-HD Audio endpoint information will be defined as an ACPI Data Table consisting of the standard ACPI Table Header followed by an endpoint descriptor for each non-HD Audio endpoint to be supported.

The platform driver retrieves ACPI configuration table by executing the following \_DSM method:

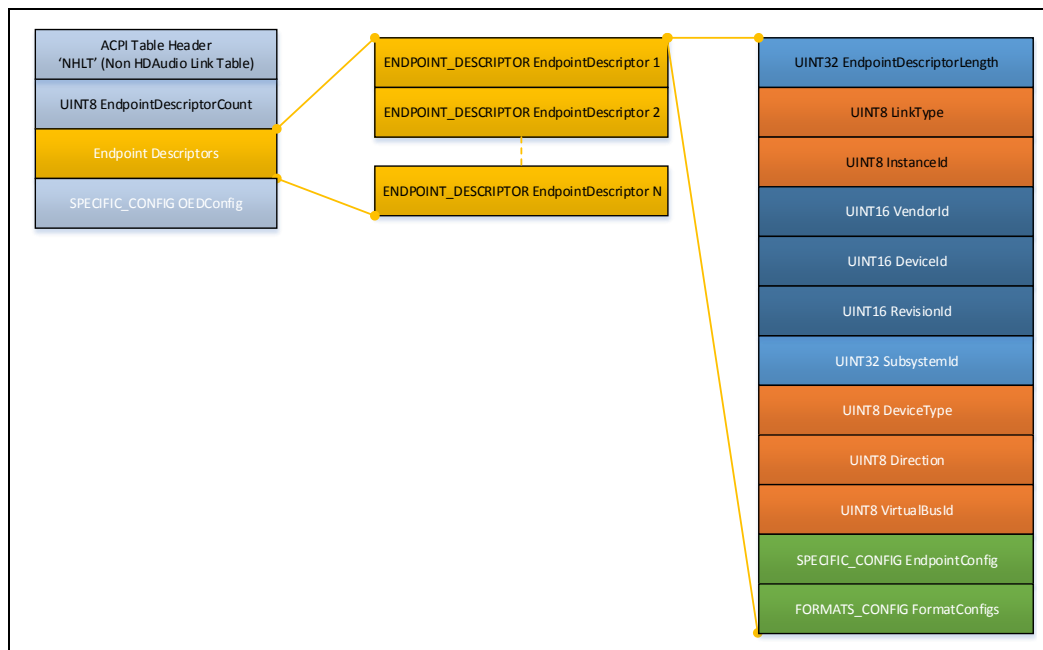
```
// Interface ID: A69F886E-6CEB-4594-A41F-7B5DCE24C553
// Function 1: Query Non HD Audio Descriptor Table
// This function is used by SST driver to discover the
// non HD Audio devices supported by the audio DSP.
Arg0 – UUID: A69F886E-6CEB-4594-A41F-7B5DCE24C553 (Buffer)
Arg1 – Revision ID: 0x01 (Integer)
// When the data structure changes this will change. Not currently
// implemented
Arg2 – Function Index: 0x01 (Integer)
Arg3 – Unused
Return – ACPI Table describing the non HD Audio links and devices supported
by the audio DSP. (Buffer)
```





The following diagram illustrates the layout of the table:

**Figure 2-1. Table Layout**



**Table 2-1. NHLT Descriptor**

Size (B)	Name	Description
4	EndpointDescriptorLength	Size of entire endpoint descriptor. This includes the size of the "EndpointDescriptorLength" field.
1 (4 bits only)	LinkType	Type of link: 0 – HD-Audio 1 – Reserved 2 – PDM 3 – SSP 4, 5, 6, 7 – Reserved for future use
1	Instance ID	Unique identification ID of connected device within Non-HD Audio Endpoint Description table (incremental number assigned to each new device). Unique to Link Type.
2	Vendor ID	Virtual device Vendor ID used for building PnP address for matching SW driver to device
2	Device ID	Virtual Device ID used for building PnP address for matching SW driver to device
2	Revision ID	Virtual device Revision ID used for building PnP address for matching SW driver to device
4	Subsystem ID	Virtual device Subsystem ID used for building PnP address for matching SW driver to device



Size (B)	Name	Description
1	Device Type	Type of device (unique to link type): <u>SSP Link:</u> 0 – BT Sideband 1 – Modem 2 – FM 3 – reserved for future use 4 – SSP Analog Codec 5-7 – reserved for future use <u>PDM Link:</u> 0 – PDM 1-7 – reserved (always set to 0)
1	Direction	Endpoint direction: 0 – Render 1 – Capture 2 – Bidirectional
1	Virtual Bus ID	Virtual Bus Line For SSP Link Type this is SSP port number. For DMIC this is always 0 because there is only one PDM link seen from SW/FW point of view. <b>Note:</b> VirtualBusID is to identify SoC side of the port while the InstanceID is to identify the device.

## 2.1 Instance ID versus Virtual Bus ID

The Instance ID has meanings for specific endpoints

	Instance ID	Virtual Bus ID
SSP	Zero based instance for devices connected to the SSP ports	SSP port number
DMIC	Always 0	Always 0

## 2.2 NHLT Element Descriptions

Note that the SPECIFIC\_CONFIG and FORMATS\_CONFIG structures are defined below.

```
typedef struct _SPECIFIC_CONFIG
{
    //Size of the "Capabilities" field in bytes.
    //Size does not include the size of field "CapabilitiesSize"
    UINT32 CapabilitiesSize;
    BYTE[] Capabilities;
} SPECIFIC_CONFIG;
```



SPECIFIC\_CONFIG is used to differentiate between TDM and I2S mode for the SSP ports and is primarily used for DMIC specifics.

```
typedef struct _FORMATS_CONFIG
{
    UINT8          FormatConfigCount;
    FORMAT_CONFIG[] FormatConfigs;
} FORMATS_CONFIG;
```

Additionally, the FORMAT\_CONFIG structure included in the FORMATS\_CONFIG structure is defined as:

```
typedef struct _FORMAT_CONFIG
{
    WAVEFORMATEXTENSIBLE Format;
    SPECIFIC_CONFIG       FormatConfig;
} FORMAT_CONFIG;
```

Where standard datatypes are defined as follows (simplified format):

```
typedef struct {
    UINT16 wFormatTag;
    UINT16 nChannels;
    UINT32 nSamplesPerSec;
    UINT32 nAvgBytesPerSec;
    UINT16 nBlockAlign;
    UINT16 wBitsPerSample;
    UINT16 cbSize;
    UINT16 wValidBitsPerSample;
    UINT32 dwChannelMask;
    GUID SubFormat;
} WAVEFORMATEXTENSIBLE;
```

**Table 2-2. WaveFormatExtensible**

Size (B)	Name	Description
2	wFormatTag	Format type of Waveform-audio, it is always 0xFFFFE for WAVEFORMATEXTENSIBLE structure.
2	wChannels	Number of channels in waveform audio data (1-mono, 2-stereo)
4	dSamplesPerSec	Waveformat audio data sample rate in samples per second
4	dAvgBytesPerSec	Waveformat audio required average data transfer rate measured in bytes per second
2	wBlockAlign	Block alignment in bytes, must be equal to value of nChannels and wBitsPerSample divided by 8 (as bits in byte)
2	wBitsPerSample	Number of bits per each sample of waveformat audio data, it may be any integer multiple of 8. This represent the container size and may be larger or equal compared to sample size.
2	cbSize	Size in bytes of extra format information appended after cbSize field, this field is set always to 22 bytes.



## Non-HD Audio Endpoint Description Table

Size (B)	Name	Description																																						
2	wValidBitsPerSample	Number of bits of precision in audio signal, may be equal to size of container as indicated in wBitsPerSample or smaller.																																						
4	dwChannelMask	<div><div>Bitmask specifying assignment of channels in the stream to speaker positions in the following manner:</div><table><tr><th>Speaker Position</th><th>Flag Bit</th></tr><tr><td>SPEAKER_FRONT_LEFT</td><td>0x1</td></tr><tr><td>SPEAKER_FRONT_RIGHT</td><td>0x2</td></tr><tr><td>SPEAKER_FRONT_CENTER</td><td>0x4</td></tr><tr><td>SPEAKER_LOW_FREQUENCY</td><td>0x8</td></tr><tr><td>SPEAKER_BACK_LEFT</td><td>0x10</td></tr><tr><td>SPEAKER_BACK_RIGHT</td><td>0x20</td></tr><tr><td>SPEAKER_FRONT_LEFT_OF_CENTER</td><td>0x40</td></tr><tr><td>SPEAKER_FRONT_RIGHT_OF_CENTER</td><td>0x80</td></tr><tr><td>SPEAKER_BACK_CENTER</td><td>0x100</td></tr><tr><td>SPEAKER_SIDE_LEFT</td><td>0x200</td></tr><tr><td>SPEAKER_SIDE_RIGHT</td><td>0x400</td></tr><tr><td>SPEAKER_TOP_CENTER</td><td>0x800</td></tr><tr><td>SPEAKER_TOP_FRONT_LEFT</td><td>0x1000</td></tr><tr><td>SPEAKER_TOP_FRONT_CENTER</td><td>0x2000</td></tr><tr><td>SPEAKER_TOP_FRONT_RIGHT</td><td>0x4000</td></tr><tr><td>SPEAKER_TOP_BACK_LEFT</td><td>0x8000</td></tr><tr><td>SPEAKER_TOP_BACK_CENTER</td><td>0x10000</td></tr><tr><td>SPEAKER_TOP_BACK_RIGHT</td><td>0x20000</td></tr></table><div><b>NOTE:</b> This field is applicable only for playback. For capture endpoints it would be provided in the microphone array data structures.</div></div>	Speaker Position	Flag Bit	SPEAKER_FRONT_LEFT	0x1	SPEAKER_FRONT_RIGHT	0x2	SPEAKER_FRONT_CENTER	0x4	SPEAKER_LOW_FREQUENCY	0x8	SPEAKER_BACK_LEFT	0x10	SPEAKER_BACK_RIGHT	0x20	SPEAKER_FRONT_LEFT_OF_CENTER	0x40	SPEAKER_FRONT_RIGHT_OF_CENTER	0x80	SPEAKER_BACK_CENTER	0x100	SPEAKER_SIDE_LEFT	0x200	SPEAKER_SIDE_RIGHT	0x400	SPEAKER_TOP_CENTER	0x800	SPEAKER_TOP_FRONT_LEFT	0x1000	SPEAKER_TOP_FRONT_CENTER	0x2000	SPEAKER_TOP_FRONT_RIGHT	0x4000	SPEAKER_TOP_BACK_LEFT	0x8000	SPEAKER_TOP_BACK_CENTER	0x10000	SPEAKER_TOP_BACK_RIGHT	0x20000
Speaker Position	Flag Bit																																							
SPEAKER_FRONT_LEFT	0x1																																							
SPEAKER_FRONT_RIGHT	0x2																																							
SPEAKER_FRONT_CENTER	0x4																																							
SPEAKER_LOW_FREQUENCY	0x8																																							
SPEAKER_BACK_LEFT	0x10																																							
SPEAKER_BACK_RIGHT	0x20																																							
SPEAKER_FRONT_LEFT_OF_CENTER	0x40																																							
SPEAKER_FRONT_RIGHT_OF_CENTER	0x80																																							
SPEAKER_BACK_CENTER	0x100																																							
SPEAKER_SIDE_LEFT	0x200																																							
SPEAKER_SIDE_RIGHT	0x400																																							
SPEAKER_TOP_CENTER	0x800																																							
SPEAKER_TOP_FRONT_LEFT	0x1000																																							
SPEAKER_TOP_FRONT_CENTER	0x2000																																							
SPEAKER_TOP_FRONT_RIGHT	0x4000																																							
SPEAKER_TOP_BACK_LEFT	0x8000																																							
SPEAKER_TOP_BACK_CENTER	0x10000																																							
SPEAKER_TOP_BACK_RIGHT	0x20000																																							
16	gSubFormat	<div>Subformat of the data indicated by GUID, it may be either PCM format {0x00000001, 0x0000, 0x0010, {0x80, 0x00, 0x00, 0xaa, 0x00, 0x38, 0x9b, 0x71}}</div> <div>Or Vendor defined subformat GUID for proprietary data subformat.</div>																																						

**Note:** All the mentioned structures are 1-byte packed.



Format of UEFI FW ACPI Description header that describes NHLT:

```
typedef struct {
    UINT32 Signature;
    UINT32 Length;
    UINT8 Revision;
    UINT8 Checksum;
    UINT8 OemId[6];
    UINT64 OemTableId;
    UINT32 OemRevision;
    UINT32 CreatorId;
    UINT32 CreatorRevision;
} EFI_ACPI_DESCRIPTION_HEADER
```

**Note:** For more information on ACPI header you can refer to Linux source at following location:

<http://lxr.free-electrons.com/source/include/acpi/actbl.h#L101>

Refer to ACPI specification, search for ACPI\_TABLE\_HEADER.

The following subsections describe low-level definitions used in ACPI Non-HD Audio Endpoint Description table for description of various endpoints connected to DSP subsystem for Intel® SST.

FORMAT\_CONFIG structure follows by SPECIFIC\_CONFIG structure and structure of SPECIFIC\_CONFIG depends on the type DeviceType. In the subsequent section the SPECIFIC\_CONFIG structure is explained for different Devices e.g. Microphones, Speakers etc.

## 2.3 Device Capabilities Structure

```
typedef struct _DEVICE_SPECIFIC_CONFIG
{
    BYTE VirtualSlot; // timeslot for multichannel transmission
    BYTE ConfigType; // 0 - basic, 1 - microphone array
} DEVICE_SPECIFIC_CONFIG;

enum eIntcConfigType
{
    eIntcConfigTypeGeneric = 0,
    eIntcConfigTypeMicArray = 1,
    eIntcConfigTypeRenderWithLoopback = 2,
    eIntcConfigTypeMultiMode = 4,

    eIntcConfigTypeMultiModeMicArray = eIntcConfigTypeMultiMode |
eIntcConfigTypeMicArray
}
```

Device specific configuration begins with the header \_DEVICE\_SPECIFIC\_CONFIG as defined above. This is generic header which is present in all types of device specific configuration (SPECIFIC\_CONFIG)



## 2.4 Multi-channel Transmission

Streams on multichannel buses (i.e.: TDM) are addressed by SW using virtual slots. FW maps link channels into virtual slots based on configuration passed by driver by reading the NHLT table.

Examples of multichannel mapping:

Capture Example.

Link Channels	0	1	2	3	4	5	6	7
Virtual Slot Assignments	3	3	0	1	2	2	2	2
Stream Channel Assignments	0	1	0	0	3	0	2	1
Stream Description	Render (HP) Loopback		Voice	Speech	Raw Capture (4-ch)			

Maximum number of channels that can be transmitted or received is restricted to 8. This is defined as “link channels” in the above table. Link channels can be grouped together to form a “stream”. This group of link channels are assigned a “virtual slot”.

Consider the capture example given above. Let’s say there is a codec connected onto I2S Port, which transmits four different streams.

1. Loopback stream.
2. Voice stream
3. Speech stream
4. Raw capture stream.

Loopback stream is transmitted in Link Slot 0 and Link Slot 1. In this example this has been given a virtual slot ID of 3. So virtual slot ID can also be looked as “Stream ID”. The “Stream ID” 3 is transmitted by I2S device in Link Slot 0 and Link Slot 1.

Maximum there can be 8 virtual slot ID, considering 8 MONO channels. In that case each virtual slot will correspond to one Link slot.

The example capture mapping is translated into following virtual slots.

Virtual Slot	Description	StreamType	Link Channels (As Ordered In Stream)
0	Voice	Voice	2
1	Speech	Speech	3
2	Raw Capture		5,7,6,4
3	Render (HP) Loopback	Loopback	0,1



Render example:

Link Channels	0	1	2	3	4	5	6	7
Virtual Slot Assignments	0	1	0	1	1	1	1	1
Stream Channel Assignment	1	1	0	0	3	2	4	5
Stream Description	Render (HP)	Render(Spk)	Render(HP)	Render (Speakers)				

Likewise consider Render case. In the above example there are two Virtual Slots or "Stream ID". Stream ID 0/Virtual Slot 0 corresponds to Link channel 0 and Link channel 2. So considering I2S link as example link, Device would receive Headphone stream in slot 0 and 2. This is bit weird but it's possible. That's why taken as example. Second Stream, Stream ID 1/Virtual Slot 1 is routed to Speakers and in this example there are 6 speakers assumed on the codec side which receives data via Link channels 1, 3, 4, 5, 6 and 7.

The example render mapping is translated into following virtual slots.

Virtual Slot	Description	Stream Type	Link Channels (As Ordered in Stream)
0	HP	-	2,0
1	Speakers	-	3,1,5,4,6,7

**Note:** One endpoint is using only one virtual bus (link line).

## 2.5 Microphone Array Device Capabilities Structure

This section defines the SPECIFIC\_CONFIG structure for microphone arrays.

```
typedef struct _MIC_ARRAY_DEVICE_SPECIFIC_CONFIG
{
    DEVICE_SPECIFIC_CONFIG DeviceConfig;
    BYTE                    ArrayType;
} MIC_ARRAY_DEVICE_SPECIFIC_CONFIG;
```

**Table 2-3. ArrayType Description**

ArrayType	Description
0xA	Linear 2-element, Small
0xB	Linear 2-element, Big
0xC	Linear 4-element, 1st geometry
0xD	Planar L-shaped 4-element
0xE	Linear 4-element, 2nd geometry
0xF	vendor defined



In many cases platform vendor would like to have its own microphone placement and in that case they can use the following structure.

```
typedef struct _VENDOR_MIC_ARRAY_DEVICE_SPECIFIC_CONFIG
{
    MIC_ARRAY_DEVICE_SPECIFIC_CONFIG DeviceConfig;
    BYTE                            NumberOfMicrophones;
    VENDOR_MIC_CONFIG                MicConfig[NumberOfMicrophones];
} VENDOR_MIC_ARRAY_DEVICE_SPECIFIC_CONFIG;

typedef struct _VENDOR_MIC_CONFIG
{
    BYTE    Type; //See Table 4
    BYTE    Panel; //See Table 5
    WORD    SpeakerPositionDistance; // in mm
    WORD    HorizontalOffset; // in mm
    WORD    VerticalOffset; // in mm
    BYTE    FrequencyLowBad; // in 5Hz units
    BYTE    FrequencyHighBad; // in 500Hz units
    SHORT    DirectionAngle; // -180 - + 180
    SHORT    ElevationAngle; // -180 - + 180
    SHORT    WorkVerticalAngleBegin; // -180 - + 180 - with step=2 deg
    SHORT    WorkVerticalAngleEnd; // -180 - + 180 - with step=2 deg
    SHORT    WorkHorizontalAngleBegin; // -180 - + 180 - with step=2 deg
    SHORT    WorkHorizontalAngleEnd; // -180 - + 180 - with step=2 deg
} VENDOR_MIC_CONFIG;
```

**Table 2-4. Microphone Type Description**

Value	Type of Microphone
0	KSMICARRAY_MICTYPE_OMNIDIRECTIONAL
1	KSMICARRAY_MICTYPE_SUBCARDIOID
2	KSMICARRAY_MICTYPE_CARDIOID
3	KSMICARRAY_MICTYPE_SUPERCARDIOID
4	KSMICARRAY_MICTYPE_HYPERCARDIOID
5	KSMICARRAY_MICTYPE_8SHAPED
6	Reserved
7	KSMICARRAY_MICTYPE_VENDORDEFINED

**Table 2-5. Panel (Location of the Microphone)**

Value	Panel (Location)
0	Top
1	Bottom
2	Left
3	Right
4	Front



Value	Panel (Location)
5	Rear
6-7	Unknown

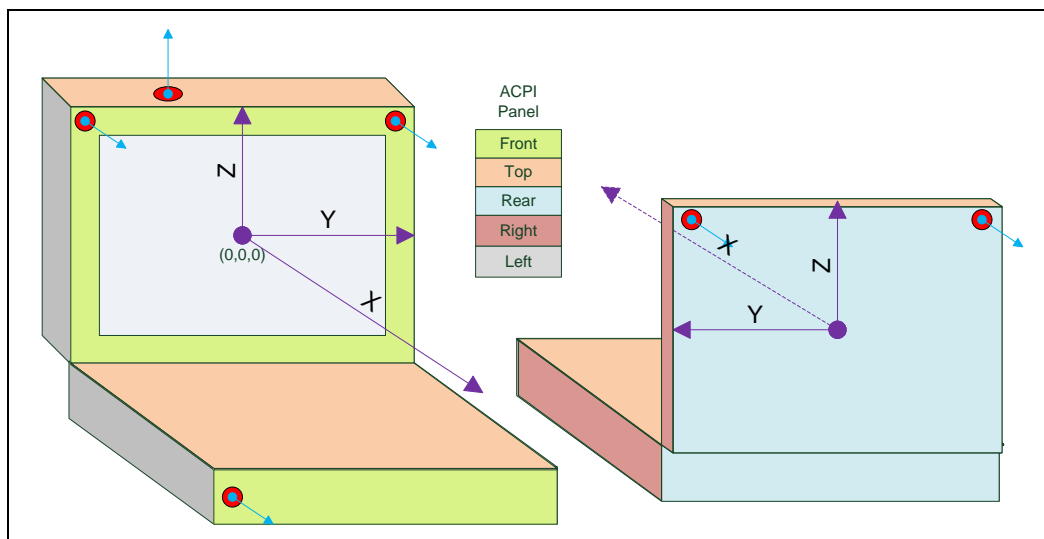
## 2.6 Microphone Array Coordinates

This section explains microphone positions.

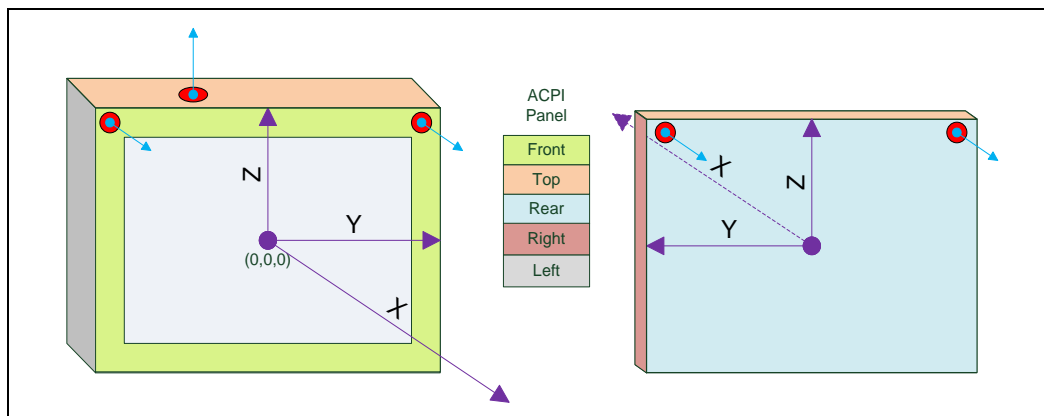
### 2.6.1 Front (Main) Microphone Array Coordinates

Below pictures show coordinate system of front (or main) microphone array in case of laptop and tablet.

**Figure 2-2 Coordinate System of Front (Main) Laptop Microphone Array**



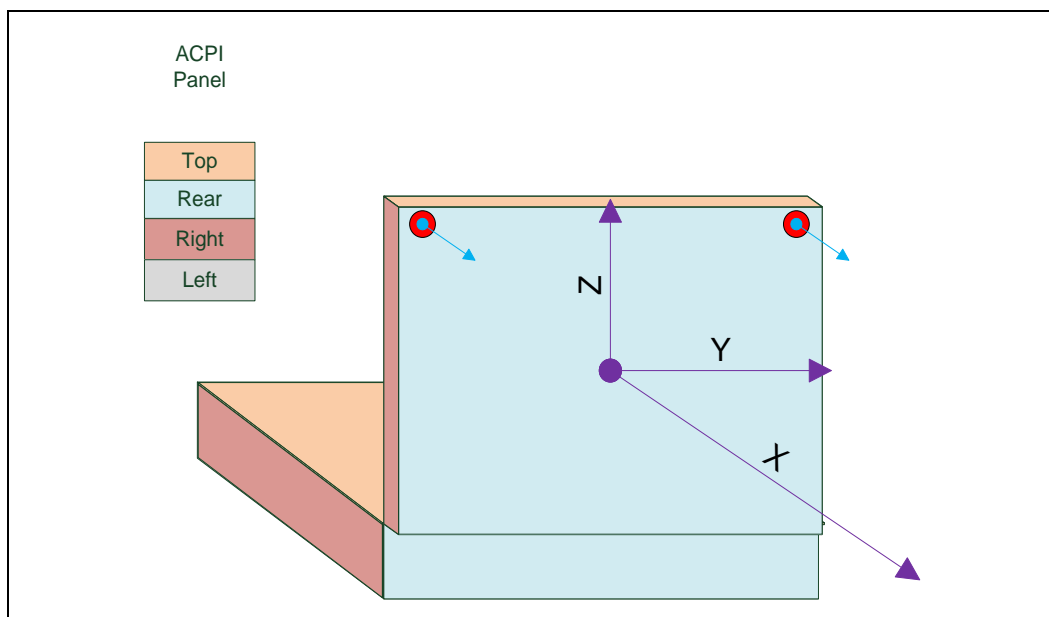
**Figure 2-3 Coordinate System of Front (Main) Tablet Microphone Array**



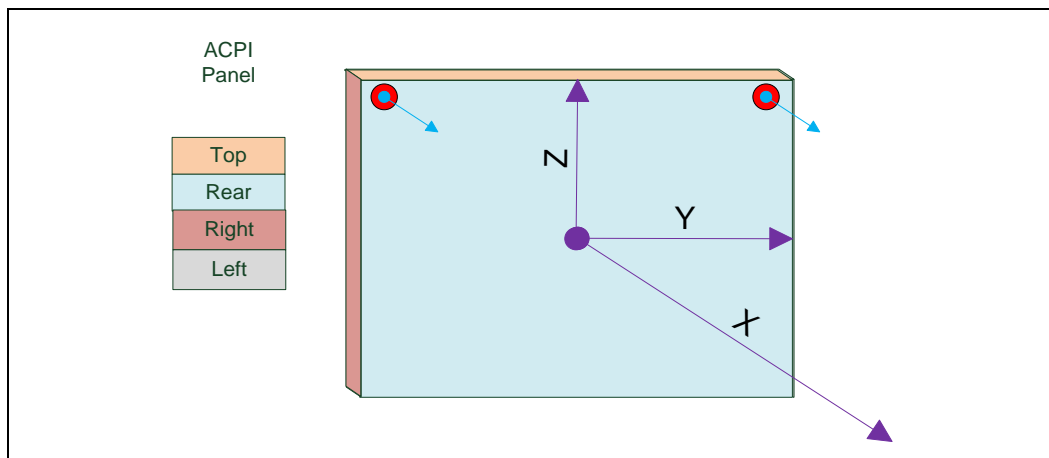
## 2.6.2 Rear (Additional) Microphone Array Coordinates

Below pictures show coordinate system of rear (additional to front) microphone array in case of laptop and tablet. Rear microphone array works with speaker in front of the rear panel.

**Figure 2-4 Coordinate System Of Rear (Additional) Laptop Microphone Array**



**Figure 2-5 Coordinate System of Rear (Additional) Tablet Microphone Array**





## 2.7 Render Device with Loopback

In case render endpoint have HW supported loopback stream (in the same stream format), capture virtual slot is provided. This can be a case where some speaker amplifier is sending the echo reference stream after doing some non-linear post processing.

```
typedef struct _RENDER_WITH_LOOPBACK_DEVICE_SPECIFIC_CONFIG
{
    DEVICE_SPECIFIC_CONFIG DeviceConfig;
    BYTE                    LoopbackVirtualSlot;
} RENDER_WITH_LOOPBACK_DEVICE_SPECIFIC_CONFIG;
```

## 2.8 Multi-mode Stream Types

Multimode stream types (depends on endpoint direction) are defined as follows:

```
/**
 * Enum defines supported OE Render stream types
 */
enum eIntcOERenderStreamType
{
    eIntcOERenderStreamTypeLoopback = 0, //loopback of render
    eIntcOERenderStreamTypeVoice,      //render with voice processing
    eIntcOERenderStreamTypeMusic,      //render - music
    eIntcOERenderStreamTypeMovie       //render with movie processing
};

/**
 * Enum defines supported OE Capture stream types
 */
enum eIntcOECaptureStreamType
{
    eIntcOECaptureStreamTypeVoice = 0, //capture with voice processing
    eIntcOECaptureStreamTypeSpeech    //capture with speech processing
};
```

## 2.9 Multi-mode Endpoint

In case endpoint have HW accelerated multi-mode streams, additional HW accelerated streams (with HW accelerated processing) are defined as follows:

```
typedef struct _MULTIMODE_SPECIFIC_CONFIG
{
    BYTE    StreamType;
    BYTE    VirtualSlot;
    BYTE    NumChannels;
} MULTIMODE_SPECIFIC_CONFIG;
```

MULTIMODE\_SPECIFIC\_CONFIG structure defines mapping of host streams other than RAW (defined by StreamType) into TDM link virtual slot. (Virtual Slot of RAW stream is already defined in VirtualSlot field of DEVICE\_SPECIFIC\_CONFIG structure.)



Valid StreamType values for each endpoint direction (render/capture) are defined in section 2.8

StreamChannels define how many channels given stream has, as other streams parameters match RAW format definition (thus link format) also stored in the same ACPI NHLT table.

```
typedef struct _MULTIMODE_STREAMS_SPECIFIC_CONFIG
{
    BYTE                      MultimodeStreamCount;
    MULTIMODE_SPECIFIC_CONFIG MultimodeStreams[MultimodeStreamCount];
} MULTIMODE_STREAMS_SPECIFIC_CONFIG;

typedef struct _MULTIMODE_DEVICE_SPECIFIC_CONFIG
{
    DEVICE_SPECIFIC_CONFIG    DeviceConfig;
    MULTIMODE_STREAMS_SPECIFIC_CONFIG MultimodeStreams;
} MULTIMODE_DEVICE_SPECIFIC_CONFIG;
```

## 2.10 Multi-mode Capture Microphone Array

In case capture mic array endpoint have HW accelerated multi-mode streams, additional HW accelerated streams are defined as follows:

```
typedef struct _MULTIMODE_MIC_ARRAY_DEVICE_SPECIFIC_CONFIG
{
    MULTIMODE_DEVICE_SPECIFIC_CONFIG DeviceConfig;
    BYTE                      ArrayType;
} MULTIMODE_MIC_ARRAY_DEVICE_SPECIFIC_CONFIG;

typedef struct _MULTIMODE_VENDOR_MIC_ARRAY_DEVICE_SPECIFIC_CONFIG
{
    MULTIMODE_MIC_ARRAY_DEVICE_SPECIFIC_CONFIG DeviceConfig;
    BYTE                      NumberOfMicrophones;
    VENDOR_MIC_CONFIG
    MicConfig[NumberOfMicrophones];
} MULTIMODE_VENDOR_MIC_ARRAY_DEVICE_SPECIFIC_CONFIG;
```





## 3 Examples

### 3.1 DMIC Connected via PDM Interface

This example provides sample values for a platform with 2 PDM DMICs that needs to be used either in 16 KHz, 2 channels, 16 bit mode or 48 KHz, 2 channels, 24bit mode.

In this case there would be one ENDPOINT\_DESCRIPTOR as shown in the table below.

**Table 3-1. ENDPOINT\_DESCRIPTOR [0]**

Fields	Value	Description
EndpointDescriptorLength	X	Length of the endpoint descriptor
LinkType	2	PDM
InstanceId	0	InstanceId is given to the device. In this case the MIC array.
VendorId	0x8086	Vendor ID
DeviceId	0xAE20	Device ID
RevisionId	1	Revision ID
SubsystemId	1	Sub System ID
DeviceType	0	Type of the device connected on this link. For PDM this is always set to 0.
Direction	1	Capture
VirtualBusId	0	PDM 0 Port

ENDPOINT\_DESCRIPTOR is followed by SPECIFIC\_CONFIG, in this case it is MIC\_ARRAY\_DEVICE\_SPECIFIC\_CONFIG as shown in the table below.

**Table 3-2. MIC\_ARRAY\_DEVICE\_SPECIFIC\_CONFIG (SPECIFIC\_CONFIG)**

Fields	Value	Description
CapabilitiesSize	3	Size of the capabilities in bytes
VirtualSlot	0	Virtual slot ID as explained in section 2.3
ConfigType	1	Microphone Array
ArrayType	0xA	Microphone Array type is Linear 2-element, Small

These microphone devices can be configured either in 16 KHz, 2 ch, 16bit or 48 KHz, 2ch, 24bit mode, so it requires two FORMAT\_CONFIG which are defined in the following table.



Table 3-3. FORMAT\_CONFIG

Fields	Value	Description
FormatConfigCount	2	Two configurations, one for 16KHz, 2 ch and 16bit and second for 48KHz, 2 ch and 24 bit mode.
FORMAT_CONFIG[0]	Please refer to structure FORMAT_CONFIG and fill the parameters as defined in the structure	WAVEFORMATEXTENSIBLE for 16KHz, 2 channels, 16bit This also contains SPECIFIC_CONFIG which contains firmware specific data structure and register settings.
FORMAT_CONFIG[1]	Please refer to structure FORMAT_CONFIG and fill the parameters as defined in the structure	WAVEFORMATEXTENSIBLE for 48KHz, 2 channels, 24bit This also contains SPECIFIC_CONFIG which contains firmware specific data structure and register settings.

## 3.2 DMIC Connected via I2S Codec

This example provides sample configuration for a platform with 4 MICS which are connected to SoC via I2S based codec. In one use case OEM wants to use only 2 MICs in I2S Mode (16KHz, 2 channels, 16 bit) and in the second use case OEM wants to use all the 4 MICs (48KHz, 4 channels, 24bit) in TDM mode.

Table 3-4. ENDPOINT\_DESCRIPTOR [0]

Fields	Value	Description
EndpointDescriptorLength	X	Length of the endpoint descriptor
LinkType	3	I2S Link
InstanceId	0	InstanceId is given to the device. In this case the MIC array.
VendorId	0x8086	Vendor ID.
DeviceId	0xAE34	Device ID.
RevisionId	1	Revision ID
SubsystemId	1	Sub System ID
DeviceType	4	Type of the device connected on this link. In this case it is analog codec.
Direction	1	Capture
VirtualBusId	2	Port number on which the device is connected, I2S Port number 2.

ENDPOINT\_DESCRIPTOR is followed by SPECIFIC\_CONFIG as shown in the following table.

**Table 3-5. SPECIFIC\_CONFIG**

Fields	Value	Description
CapabilitiesSize	0	Size of the capabilities

In this case it is assumed that it is I2S mode and it has pre-defined slot for left channel and right channel and so there is no need of device specific configuration.

**Table 3-6. FORMAT\_CONFIG**

Fields	Value	Description
FormatConfigCount	1	Only one format configuration
FORMAT_CONFIG[0]		WAVEFORMATEXTENSIBLE for 16KHz, 2 channels, 16bit This also contains SPECIFIC_CONFIG which contains firmware specific data structure and register settings.

For the second use case, where it is using all the 4 microphones in TDM mode with 48Khz, 24bit mode, we need to define new endpoint descriptor. This is because the geometric information for the placement of microphone in both the usecases are different.

**Table 3-7. ENDPOINT\_DESCRIPTOR[1]**

Fields	Value	Description
EndpointDescriptorLength	X	Length of the endpoint descriptor
LinkType	3	I2S Link
InstanceId	0	InstanceId is given to the device. In this case the MIC array.
VendorId	0x8086	Vendor ID of the device.
DeviceId	0xAE34	Device ID of the device.
RevisionId	1	Revision ID
SubsystemId	1	Sub System ID
DeviceType	4	Type of the device connected on this link. In this case it is analog codec.
Direction	1	Capture
VirtualBusId	2	Port number on which the device is connected, I2S Port number 2.

**Table 3-8. VENDOR\_MIC\_ARRAY\_DEVICE\_SPECIFIC\_CONFIG (SPECIFIC\_CONFIG)**

Fields	Value	Description
CapabilitiesSize	3	Size of the capabilities
VirtualSlot	0	Virtual slot ID as explained in section 2.3



Fields	Value	Description
ConfigType	1	Microphone Array
ArrayType	0xF	Microphone Array type is Vendor defined
VENDOR_MIC_CONFIG	Based on vendor preference	Vendor can fill in VENDOR_MIC_CONFIG structure based on the platform configuration

### 3.3 BT Sideband Device Connected via I2S Port

This example provides sample configuration for a BT device connected to SoC via I2S port. In this case there are two ENDPOINT\_DESCRIPTOR created, one for Render and second for Capture. The reason for creating two separate ENDPOINT\_DESCRIPTOR instances is to specify separate configuration for render and capture.

**Table 3-9. ENDPOINT\_DESCRIPTOR [0] (Render)**

Fields	Value	Description
EndpointDescriptorLength	X	Length of the endpoint descriptor
LinkType	3	I2S
InstanceId	0	InstanceId is given to the device. In this case example since there is only one I2S device this is 0.
VendorId	0x8086	Vendor ID
DeviceId	0xAE30	Device ID
RevisionId	1	Revision ID
SubsystemId	1	Sub System ID
DeviceType	0	Type of the device connected on this link. 0 for BT side band device.
Direction	0	Playback/Render
VirtualBusId	1	I2S 1 Port

ENDPOINT\_DESCRIPTOR is followed by SPECIFIC\_CONFIG. Since there is no real specific configuration required in this case the SPECIFIC\_CONFIG is empty with zero size.

**Table 3-10. SPECIFIC\_CONFIG**

Fields	Value	Description
CapabilitiesSize	0	Size of the capabilities

SPECIFIC\_CONFIG is followed by FORMAT\_CONFIG which basically contains the PCM configuration as 8 KHz, 1 channel, 16 bit for BT HFP profile.



**Table 3-11. FORMAT\_CONFIG**

Fields	Value	Description
FormatConfigCount	1	Only one format configuration
FORMAT_CONFIG[0]		WAVEFORMATEXTENSIBLE for 8KHz, 1 channels, 16bit This also contains SPECIFIC_CONFIG which contains firmware specific data structure and register settings.

**Table 3-12. ENDPOINT\_CONFIGURATION [1] (Capture)**

Fields	Value	Description
EndpointDescriptorLength	X	Length of the endpoint descriptor
LinkType	3	I2S
InstanceId	0	InstanceId is given to the device. In this case example since there is only one I2S device this is 0.
VendorId	0x8086	Vendor ID
DeviceId	0xAE30	Device ID
RevisionId	1	Revision ID
SubsystemId	1	Sub System ID
DeviceType	0	Type of the device connected on this link. 0 for BT side band device.
Direction	1	Capture
VirtualBusId	1	I2S 1 Port

ENDPOINT\_DESCRIPTOR is followed by SPECIFIC\_CONFIG as shown in Table 3-14. FORMAT\_CONFIG. Since there is no real specific configuration required in this case the SPECIFIC\_CONFIG is empty with zero size.

**Table 3-13. SPECIFIC\_CONFIG**

Fields	Value	Description
CapabilitiesSize	0	Size of the capabilities

SPECIFIC\_CONFIG is followed by FORMAT\_CONFIG which basically contains the PCM configuration as 8 KHz, 1 channel, 16 bit for BT HFP profile.



Table 3-14. FORMAT\_CONFIG

Fields	Value	Description
FormatConfigCount	1	Only one format configuration
FORMAT_CONFIG[0]		WAVEFORMATEXTENSIBLE for 8KHz, 1 channels, 16bit This also contains SPECIFIC_CONFIG which contains firmware specific data structure and register settings.

### 3.4 SSP (TDM Mode) Device ACPI Configuration

This example provides sample configuration for an I2S based codec connected to SoC via I2S interface, communicating in multi slot TDM Mode.

Between codec and SoC there are basically two render streams, one for headset and one for speakers. In addition to that there are two capture streams. The first capture stream is receiving data from DMIC0 and DMIC1, and second capture stream is receiving data from DMIC2 and DMIC3.

Headset stream is getting transmitted on VirtualSlot0 and Speaker stream is getting transmitted on VirtualSlot1. This information is represented by ENDPOINT\_CONFIGURATION[0] and [1] respectively.

Similarly DMIC(0,1) stream is getting received on VirtualSlot0 and DMIC(2,3) stream is getting received on VirtualSlot1. This information is represented by ENDPOINT\_CONFIGURATION[2] and [3] respectively.

Table 3-15. ENDPOINT\_CONFIGURATION[0] (Headset)

Fields	Value	Description
EndpointDescriptorLength	X	Length of the endpoint descriptor
LinkType	3	I2S
InstanceId	0	InstanceId is given to the device. In this case example since there is only one I2S device this is 0.
VendorId	0x8086	Example Vendor ID
DeviceId	0xAE34	Example Device ID
RevisionId	1	Revision ID
SubsystemId	1	Sub System ID
DeviceType	4	Type of the device connected on this link. 4 for analog codec device.
Direction	0	Playback/Render
VirtualBusId	1	I2S 1 Port



Table 3-16. DEVICE\_SPECIFIC\_CONFIG (Headset)

Fields	Value	Description
CapabilitiesSize	2	Size of the capabilities
VirtualSlot	0	Virtual slot ID as explained in section 2.3 In this case the codec headset receives data from Virtual slot 0, which corresponds to Link Slot 0 and 1.
ConfigType	0	Basic configuration

Table 3-17. FORMAT\_CONFIG (Headset)

Fields	Value	Description
FormatConfigCount	1	48Khz, 2 ch and 24 bit mode.
FORMAT_CONFIG[0]	Please refer to structure FORMAT_CONFIG and fill the parameters as defined in the structure	WAVEFORMATEXTENSIBLE for 48Khz, 2 ch and 24 bit mode. This also contains SPECIFIC_CONFIG which contains firmware specific data structure and register settings. Virtual Slot 0 in the DEVICE_SPECIFIC_CONFIG would have a corresponding entry in the blob (SPECIFIC_CONFIG) of FORMAT_CONFIG which would indicate to transmit it in Slot 0 and Slot 1 of the Link.

Table 3-18. ENDPOINT\_CONFIGURATION [1] (Speakers)

Fields	Value	Description
EndpointDescriptorLength	X	Length of the endpoint descriptor
LinkType	3	I2S
InstanceId	0	InstanceId is given to the device. In this case example since there is only one I2S device this is 0.
VendorId	0x8086	Example Vendor ID
DeviceId	0xAE34	Example Device ID
RevisionId	1	Revision ID
SubsystemId	1	Sub System ID
DeviceType	4	Type of the device connected on this link. 4 for analog codec device.
Direction	0	Playback/Render
VirtualBusId	1	I2S 1 Port



Table 3-19. DEVICE\_SPECIFIC\_CONFIG (Speakers)

Fields	Value	Description
CapabilitiesSize	2	Size of the capabilities
VirtualSlot	1	Virtual slot ID as explained in section 2.3 In this case the codec speaker receives data from Virtual slot 1, which corresponds to Link Slot 2 and 3.
ConfigType	0	Basic configuration

Table 3-20. FORMAT\_CONFIG (Speakers)

Fields	Value	Description
FormatConfigCount	1	48Khz, 2 ch and 24 bit mode.
FORMAT_CONFIG[0]	Please refer to structure FORMAT_CONFIG and fill the parameters as defined in the structure	WAVEFORMATEXTENSIBLE for 48Khz, 2 ch and 24 bit mode. This also contains SPECIFIC_CONFIG which contains firmware specific data structure and register settings. Virtual Slot 1 in the DEVICE_SPECIFIC_CONFIG would have a corresponding entry in the blob (SPECIFIC_CONFIG) of FORMAT_CONFIG which would indicate to transmit it in Slot 2 and Slot 3 of the Link.

Next is to define the capture endpoint.

Table 3-21. ENDPOINT\_CONFIGURATION [2] (DMIC0,1)

Fields	Value	Description
EndpointDescriptorLength	X	Length of the endpoint descriptor
LinkType	3	I2S
InstanceId	0	InstanceId is given to the device. In this case example since there is only one I2S device this is 0.
VendorId	0x8086	Example Vendor ID
DeviceId	0xAE34	Example Device ID
RevisionId	1	Revision ID
SubsystemId	1	Sub System ID
DeviceType	4	Type of the device connected on this link. 4 for analog codec device.
Direction	1	Capture
VirtualBusId	1	I2S 1 Port

**Table 3-22. DEVICE\_SPECIFIC\_CONFIG (DMIC0,1)**

Fields	Value	Description
CapabilitiesSize	2	Size of the capabilities
VirtualSlot	0	Virtual slot ID as explained in section 2.1 In this case the codec sends DMIC(0,1) data on Virtual slot 0, which corresponds to Link Slot 0 and 1.
ConfigType	0	Basic configuration

In this particular example, the 2 microphones are not really treated as Microphone array and so the ConfigType is set as 0. Some OEM may want to treat it as microphone array and can change this field accordingly.

**Table 3-23. FORMAT\_CONFIG (DMIC0,1)**

Fields	Value	Description
FormatConfigCount	1	48Khz, 2 ch and 24 bit mode.
FORMAT_CONFIG[0]	Please refer to structure FORMAT_CONFIG and fill the parameters as defined in the structure	WAVEFORMATEXTENSIBLE for 48Khz, 2 ch and 24 bit mode. This also contains SPECIFIC_CONFIG which contains firmware specific data structure and register settings. Virtual Slot 0 in the DEVICE_SPECIFIC_CONFIG would have a corresponding entry in the blob (SPECIFIC_CONFIG) of FORMAT_CONFIG which would indicate to receive it in Slot 0 and Slot 1 of the Link.

Endpoint configuration for the second capture stream.

**Table 3-24. ENDPOINT\_CONFIGURATION [3] (DMIC2,3)**

Fields	Value	Description
EndpointDescriptorLength	X	Length of the endpoint descriptor
LinkType	3	I2S
InstanceId	0	InstanceId is given to the device. In this case example since there is only one I2S device this is 0.
VendorId	0x8086	Example Vendor ID
DeviceId	0xAE34	Example Device ID
RevisionId	1	Revision ID
SubsystemId	1	Sub System ID
DeviceType	4	Type of the device connected on this link. 4 for analog codec device.
Direction	1	Capture
VirtualBusId	1	I2S 1 Port



Table 3-25. DEVICE\_SPECIFIC\_CONFIG (DMIC2,3)

Fields	Value	Description
CapabilitiesSize	2	Size of the capabilities
VirtualSlot	1	Virtual slot ID as explained in section 2.1 In this case the codec sends DMIC(2,3) data on Virtual slot 1, which corresponds to Link Slot 2 and 3.
ConfigType	0	Basic configuration

Table 3-26. FORMAT\_CONFIG (DMIC2,3)

Fields	Value	Description
FormatConfigCount	1	48Khz, 2 ch and 24 bit mode.
FORMAT_CONFIG[0]	Please refer to structure FORMAT_CONFIG and fill the parameters as defined in the structure	WAVEFORMATEXTENSIBLE for 48Khz, 2 ch and 24 bit mode. This also contains SPECIFIC_CONFIG which contains firmware specific data structure and register settings. Virtual Slot 1 in the DEVICE_SPECIFIC_CONFIG would have a corresponding entry in the blob (SPECIFIC_CONFIG) of FORMAT_CONFIG which would indicate to receive it in Slot 2 and Slot 3 of the Link.

**Note:** It is not compulsory to split the 4 microphones into two separate streams. Depending on the requirement some vendor can have just one single capture stream receiving data from all the 4 DMICs. In that case there can be only one ENDPOINT\_CONFIGURATION entry required on the capture side. We can still have 2 separate streams on the playback side.

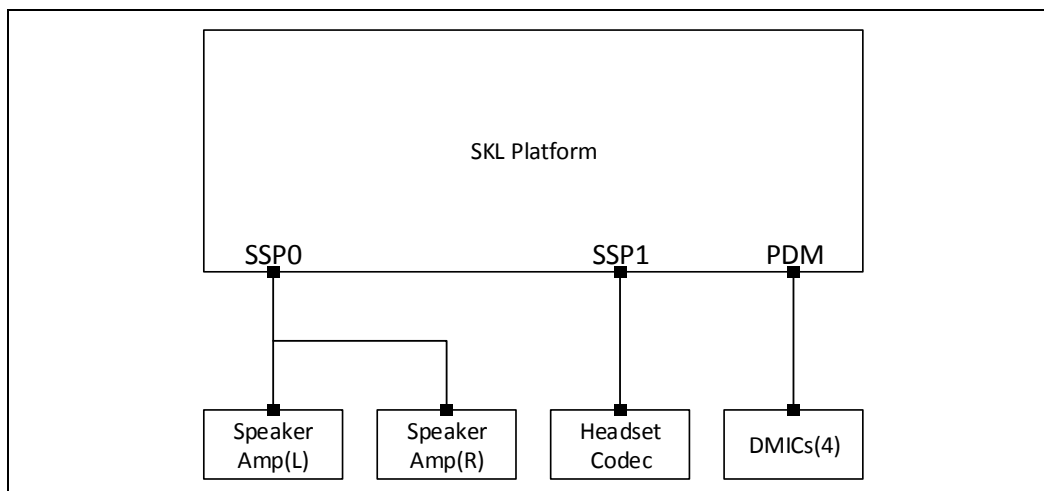
## 3.5 Sample NHLT Example

This section explains sample NHLT configuration by taking a hypothetical SKL reference platform. This section should be taken as example only and may not match with a direct platform.

### 3.5.1 Platform Connectivity

The figure below shows platform connectivity for the SKL reference platform.

Figure 3-1. SKL Reference Platform Connectivity



SSP0 is connected to speaker amplifiers and configured in TDM mode, 2 slots

SSP1 is connected to headset and is configured in I2S mode

### 3.5.2 NHLT Table

There are 4 Endpoint Descriptors are required for based on the above configuration. Endpoint configurations are given in Table 3-27.

Table 3-27. NHLT Table

Fields	Value	Description
ACPI Table Header	See description	Please fill the values as per structure EFI_ACPI_DESCRIPTION_HEADER.
EndpointDescriptorCount	4	Number of endpoints described in NHLT
EndpointDescriptor[0]	Refer Table 3-28. ENDPOINT_DESCRIPTOR [0] (DMICs)	Endpoint configuration for DMICs. Even though there are four DMICs they are exposed as a single endpoints because DMICs cannot be configured independently with different parameters.
EndpointDescriptor[1]	Refer Table 3-31.	Endpoint configuration for Speaker Amplifiers, even though there are two different speaker amplifiers it is exposed as a single endpoint because they are not expected to be two different endpoints.
EndpointDescriptor[2]	Refer Table 3-34.	Endpoint configuration for Headset codec. Playback Endpoint configuration.
EndpointDescriptor[3]	Refer Table 3-29	Endpoint configuration for Headset codec. Capture Endpoint configuration.
CapabilitiesSize	0	SPECIFIC_CONFIG OEDConfig field present should have size of 0.



Table 3-28. ENDPOINT\_DESCRIPTOR [0] (DMICs)

Fields	Value	Description
EndpointDescriptorLength	X	Size of entire endpoint descriptor, this includes the size of the "EndpointDescriptorLength" field.
LinkType	2	PDM
InstanceId	0	There is only PDM Link and all the microphones are seen as single endpoint and so only single InstanceId.
VendorId	0x8086	Virtual device Vendor ID used for building PnP address for matching Windows SW driver to device.
DeviceId	0xae20	Virtual device ID used for building PnP address for matching Windows SW driver to device.
RevisionId	1	Virtual device Revision ID used for building PnP address for matching SW driver to device.
SubsystemId	1	Virtual device Subsystem ID used for building PnP address for matching SW driver to device.
DeviceType	0	PDM device. This is unique to LinkType.
Direction	1	Capture
VirtualBusId	0	There is only 1 PDM link, even though there are 4 different microphones connected, they are seen as Single endpoint.

Table 3-29. SPECIFIC\_CONFIG

Fields	Value	Description
CapabilitiesSize	0	No extra capabilities for the microphone.

Table 3-30. FORMATS\_CONFIG

Fields	Value	Description
FormatConfigCount	2	Two types configurations for DMIC endpoints.
FORMAT_CONFIG[0]	Please refer to structure FORMAT_CONFIG and fill the parameters as defined in the structure	16Khz, 24bit, 2 channels
FORMAT_CONFIG[1]	Please refer to structure FORMAT_CONFIG and fill the parameters as defined in the structure	48Khz, 24bit, 4 channels

**Note:** There are two Format Configuration required based on the SKL reference use cases.

FORMAT\_CONFIG[0] – This would be used in the capture use case where firmware would be doing the 24 bit to 16bit conversion. However the blob would configure the PDM blocks into 24bit mode.

FORMAT\_CONFIG [1] – This would be used in two scenarios. One for the regular capture use case with 4 channel, 24bit, 4 channel mode and second for alternate capture use case simultaneously. The blob is same for the PDM blocks for 16KHz and





24KHz configuration. In all cases whenever required the 24bit to 16bit conversion is done by firmware and that's why the blob configures the PDM interface always in 24bit mode.

When the LinkType is PDM, Driver ignores the "frequency" field to search the table. The PDM interface always have same blob settings for 16 and 48KHz. This reduces the number of configurations and the size of the table.

**Table 3-31. ENDPOINT\_DESCRIPTOR [1] (Speaker Amplifiers - Playback)**

Fields	Value	Description
EndpointDescriptorLength	X	Size of entire endpoint descriptor, this includes the size of the "EndpointDescriptorLength" field.
LinkType	3	SSP Link
InstanceId	0	This is first instance of the device connected to SSP Link so it is set to 0. InstanceId is unique per LinkType.
VendorId	0x8086	Virtual device Vendor ID used for building PnP address for matching Windows SW driver to device.
DeviceId	0xae34	Virtual device ID used for building PnP address for matching Windows SW driver to device.
RevisionId	1	Virtual device Revision ID used for building PnP address for matching SW driver to device.
SubsystemId	1	Virtual device Subsystem ID used for building PnP address for matching SW driver to device.
DeviceType	4	Device type is Analog Codec
Direction	0	Playback / Render
VirtualBusId	0	SSP0. Codec is connected to SSP0

**Table 3-32. DEVICE\_SPECIFIC\_CONFIG (Speaker Amplifiers – Playback)**

Fields	Value	Description
CapabilitiesSize	2	2 bytes for the capabilities.
VirtualSlot	0	There is only one virtual slot in the TDM mode. There are two Link slots in the TDM mode but it is seen as a single stream.
ConfigType	0	Basic

**Table 3-33. FORMATS\_CONFIG (Speaker Amplifiers – Playback)**

Fields	Value	Description
FormatConfigCount	1	Only one configuration for the Speaker Amplifiers.
FORMAT_CONFIG[0]	Please refer to structure FORMAT_CONFIG and fill the parameters as defined in the structure	48KHz, 24bit, 2 channels in TDM mode.



Table 3-34. ENDPOINT\_DESCRIPTOR [2] (Headset - Playback)

Fields	Value	Description
EndpointDescriptorLength	X	Size of entire endpoint descriptor, this includes the size of the "EndpointDescriptorLength" field.
LinkType	3	SSP
InstanceId	1	This is new device connected to SSP Link so it InstanceId is incremented from 0 to 1. InstanceId is unique per LinkType.
VendorId	0x8086	Virtual device Vendor ID used for building PnP address for matching SW driver to device. This is mainly used for loading Windows MINI port drivers.
DeviceId	0xae34	Virtual device Device ID used for building PnP address for matching Windows SW driver to device.
RevisionId	1	Virtual device Revision ID used for building PnP address for matching SW driver to device.
SubsystemId	1	Virtual device Subsystem ID used for building PnP address for matching SW driver to device.
DeviceType	4	Device type is Analog Codec
Direction	0	Playback / Render
VirtualBusId	1	SSP1, Headset Codec is connected to SSP1

Table 3-35. SPECIFIC\_CONFIG (Headset - Playback)

Fields	Value	Description
CapabilitiesSize	0	No extra capabilities for the headset codec configured in I2S mode.

Table 3-36. FORMATS\_CONFIG (Headset - Playback)

Fields	Value	Description
FormatConfigCount	1	Only one configuration for the codec.
FORMAT_CONFIG[0]	Please refer to structure FORMAT_CONFIG and fill the parameters as defined in the structure	48Khz, 24bit, 2 channels in I2S mode.

Table 3-37. ENDPOINT\_DESCRIPTOR [3] (Headset - Capture)

Fields	Value	Description
EndpointDescriptorLength	X	Size of entire endpoint descriptor, this includes the size of the "EndpointDescriptorLength" field.
LinkType	3	SSP
InstanceId	1	InstanceId is already incremented in the Endpoint Descriptor[2] – Headset Playback and it's for the same device so the value remains 1 here.
VendorId	0x8086	Virtual device Vendor ID used for building PnP address for matching SW driver to device. This is mainly used for



Fields	Value	Description
		loading Windows MINI port drivers.
DeviceId	0xae34	Virtual device Device ID used for building PnP address for matching Windows SW driver to device.
RevisionId	1	Virtual device Revision ID used for building PnP address for matching SW driver to device.
SubsystemId	1	Virtual device Subsystem ID used for building PnP address for matching SW driver to device.
DeviceType	4	Device type is Analog Codec
Direction	1	Capture
VirtualBusId	1	SSP1, headset Codec is connected to SSP1

**Table 3-38. SPECIFIC\_CONFIG (Headset - Capture)**

Fields	Value	Description
CapabilitiesSize	0	No extra capabilities for the headset codec configured in I2S mode.

**Table 3-39: FORMATS\_CONFIG (Headset - Capture)**

Fields	Value	Description
FormatConfigCount	1	Only one configuration for the codec.
FORMAT_CONFIG[0]	Please refer to structure FORMAT_CONFIG and fill the parameters as defined in the structure.	48Khz, 24bit, 2 channels in I2S mode.

§ §