High Definition Audio Specification

Revision 1.0

April 15, 2004

Revision History

Revision	Purpose	Date
1.0	Initial Release	April 15, 2004

Legal Notice

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Intel disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein, except that a license is hereby granted to copy and reproduce this specification for internal use only. Intel assumes no responsibility for any errors contained in this document and has no liabilities or obligations for any damages arising from or in connection with the use of this document.

Intel may make changes to specifications, product descriptions, and plans at any time, without notice.

Intel may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

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

Copyright © 2003-2004, Intel Corporation. All rights reserved.

Contents

Figures

Tables

1 Introduction

1.1 Scope and Layout of This Document

This document is divided into seven chapters.

Chapter 1 is an introduction to the specification. It contains a high level overview of the specification layout, goals, and non-goals.

Chapter 2 provides an introduction to the High Definition Audio architecture.

Chapter 3 describes the details of a High Definition Audio controller register set, as well as the data structures used in programming the controller. Software programmers and controller hardware designers will find this chapter of interest.

Chapter 4 is a software programming reference designed to aid in the development of High Definition Audio compliant software, including controller and codec programming. This chapter is of primary interest to those writing High Definition Audio compliant software or to readers looking for more detail about the functionality of the hardware.

Chapter 5 describes the High Definition Audio physical link protocol. Designers of controller and codec hardware will find this chapter of interest.

Chapter 6 describes the physical link electrical requirements necessary for interfacing High Definition Audio controllers with codecs on the link. Designers of controller and codec hardware will find this chapter of interest.

Chapter 7 describes the High Definition Audio codec architecture. Information on codec architectural building blocks, as well as the software programming model, is provided. Codec hardware designers and software programmers will find this chapter of interest.

1.2 Motivation and Goals

The primary goals of the High Definition Audio (HD Audio) specification are to describe an infrastructure to support high quality audio in a PC environment. The specification includes the definition of the controller register set, the physical link description, the codec programming model, and codec architectural components.

The High Definition Audio specification primarily focuses on audio functionality for purposes of architectural description. Other codec types are implementable within the High Definition Audio architecture. In particular, voice band modems codecs (v.92, for instance) are supported, as the modem data rates and types of data are typically a subset of the audio data rates and types. See Section 7.2.2.2 for more information about modem support. In general, the High Definition Audio specification does not prohibit the implementation of other codec types provided that they comply with all codec requirements laid out in Chapters 3 through 7 of this specification.

1.2.1 AC‘97 Compatibility

The High Definition Audio architecture was not developed with the intent of being backward compatible with AC’97. Unlike AC’ 97, the primary goal of the High Definition Audio Architecture is to develop a uniform programming interface and to provide fundamental flexibility and capabilities beyond those supported by AC’97. Therefore, the High Definition Audio architecture and specification must be considered entirely a separate specification from AC’97, implying no backward compatibility. Specifically, link protocol and operation between these two specifications is not compatible. AC’97 and High Definition Audio codecs cannot be mixed on the same link or behind the same controller.

1.2.2 Feature List

- Support for 15 input and 15 output streams at a time
- Extensive support for scalability in controller, link, and codec design to optimize for cost, performance, or features
- Sample rate support ranging from 6 kHz to 192 kHz.
- Support for 8-, 16-, 20-, 24-, and 32-bit sample resolution per stream.
- Up to 16 channels per stream.
- 48-Mbps outbound link transfer rate per **SDO**.
- 24-Mbps inbound transfer rate per **SDI**.
- Support for striping on optional higher order **SDO** link pins to double or quadruple available outbound bandwidth.
- Support for multi-**SDI** codecs to increase available inbound link bandwidth.
- Codec architecture is fully discoverable, allowing for codec design flexibility.
- Audio codecs, modem codecs, and vendor defined codecs are all supported.
- Command/Response codec communication mechanism for extensibility and flexibility.
- Support for system wake generation from all codecs types.
- Support for codec interrupt generation through Unsolicited Responses.
- Extensive, fine grained power management control in the codec.
- Industry standard 48-pin QFP package and pinout for codec.
- Audio codecs support advanced jack detection and jack sensing for device discoverability and jack retasking.

1.2.3 Related Documents

PCI Local Bus Specification, Rev. 2.3

PCI Power Management Interface Specification, Rev. 1.2

PCI Express Base Specification, Rev. 1.0a*

Advanced Configuration and Power Interface Specification, Rev. 2.0

2 Architecture Overview

The purpose of this chapter is to introduce terminology specific to the High Definition Audio architecture, which will be used throughout this specification, and to provide a qualitative introduction to or theory of operation for the High Definition Audio architecture. This conceptual overview should give the reader a foundation for ease of navigating through the interface and syntactical details delivered in the balance of the specification. It is not the intent of this chapter to provide any syntactic, timing, or otherwise quantitative definitions.

2.1 Hardware System Overview

The hardware building blocks of the High Definition Audio architecture are shown in Figure 1.

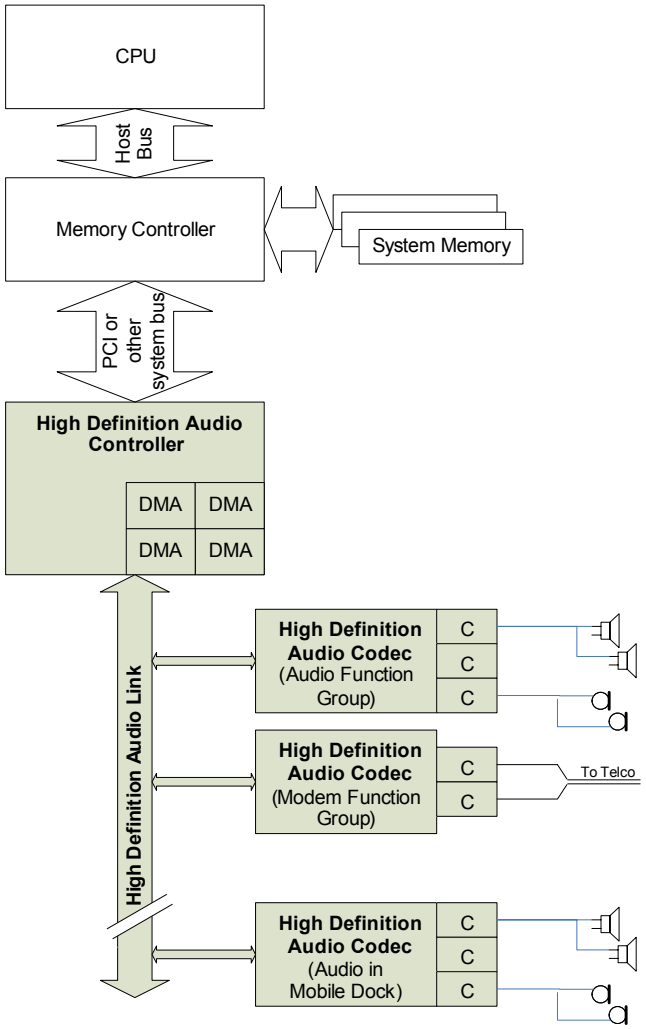

Figure 1. High Definition Audio Architecture Block Diagram

Controller: The High Definition Audio controller is a bus mastering I/O peripheral, which is attached to system memory via PCI or other typical PC peripheral attachment host interface. It contains one or more DMA engines, each of which can be set up to transfer a single audio "stream" to memory from the codec or from memory to the codec depending on the DMA type. The controller implements all the memory mapped registers that comprise the programming interface as defined in Section 3.3.

Link: The controller is physically connected to one or more codecs via the High Definition Audio Link. The link conveys serialized data between the controller and the codecs. It is optimized in both bandwidth and protocol to provide a highly cost effective attach point for lost cost codecs. The link also distributes the sample rate time base, in the form of a link *bit clock (BCLK)*, which is generated by the controller and used by all Codecs. The link protocol supports a variety of sample rates and sizes under a fixed data transfer rate.

Codec: One or more codecs connect to the link. A codec extracts one or more audio streams from the time multiplexed link protocol and converts them to an output stream through one or more converters (marked "C"). A converter typically converts a digital stream into an analog signal (or vise versa), but may also provide additional support functions of a modem and attach to a phone line, or it may simply de-multiplex a stream from the link and deliver it as a single (un-multiplexed) digital stream, as in the case of S/PDIF. The number and type of converters in a codec, as well as the type of jacks or connectors it supports, depend on the codec's intended function. The codec derives its sample rate clock from a clock broadcast (BCLK) on the link. High Definition Audio Codecs are operated on a standardized command and control protocol as defined in Section 4.4.

Acoustic Device: These devices include speakers, headsets, and microphones, and the specifications for them are outside the scope of this specification, except for discovery capabilities such as defined in Section 7.3.3.15.

Packaging Alternatives: Figure 1 suggests that codecs can be packaged in a variety of ways, including integration with the controller, permanent attachment on the motherboard, modular ("add-in") attachment, or included in a separate subsystem such as a mobile docking station. In general the electrical extensibility and robustness of the link is the limiting factor in packaging options. This specification does not define or standardize packaging options beyond the standardized footprint of a codec as defined in Section 7.4.1 of this specification.

2.2 Streams and Channels

The High Definition Audio architecture introduces the notion of streams and channels for organizing data that is to be transmitted across the High Definition Audio link. A *stream* is a logical or virtual connection created between a system memory buffer(s) and the codec(s) rendering that data, which is driven by a single DMA channel through the link. A stream contains one or more related components or *channels* of data, each of which is dynamically bound to a single converter in a codec for rendering. For example, a simple stereo stream would contain two channels; left and right. Each sample point in that stream would contain two samples: L and R. The samples are packed together as they are represented in the memory buffer or transferred over the link, but each are bound to a separate D-to-A converter in the codec.

Figure 2 illustrates several important concepts. First, a stream is either output or input. Output streams are broadcast and may be bound to more than one codec; e.g., stream #3 might be a two-channel (stereo) stream rendered by both Codec-A on a headset and by Codec-C on speakers. Input streams may be bound to only a single codec; e.g., stream #2 contains the single channel – the input side of a modem.

Each active stream must be connected through a DMA engine in the controller. If a DMA engine is not available, a stream must remain inactive until one becomes available; e.g., stream #4 in this example (presumably the one bound to the headset microphone) is not connected to a DMA engine and is therefore inactive.

As a general rule, all channels within a stream must have the same sample rate and same sample size.

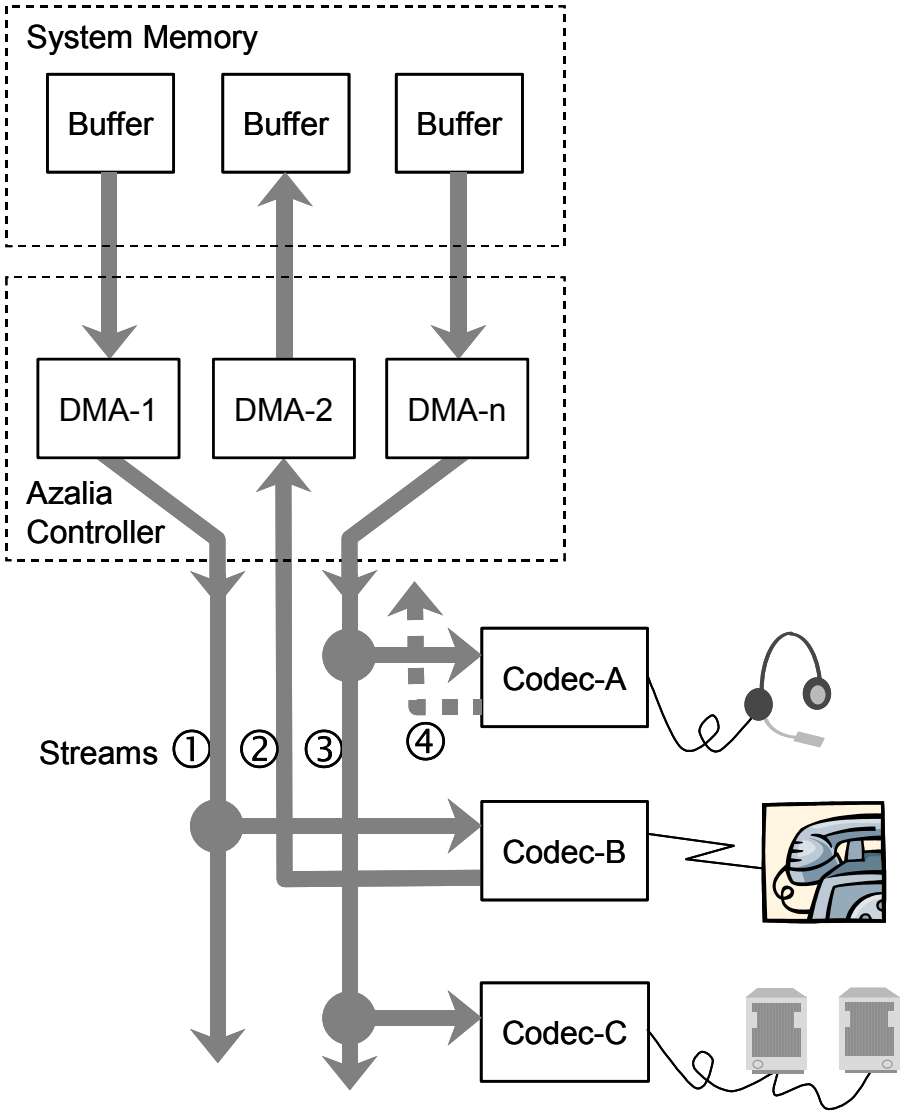

Figure 2. Streams

Figure 3 shows how streams and channels are transferred on the link. Each input or output signal in the link transmits a series of packets or *frames*. A new frame starts exactly every 20.83 µs, corresponding to the common 48-kHz sample rate.

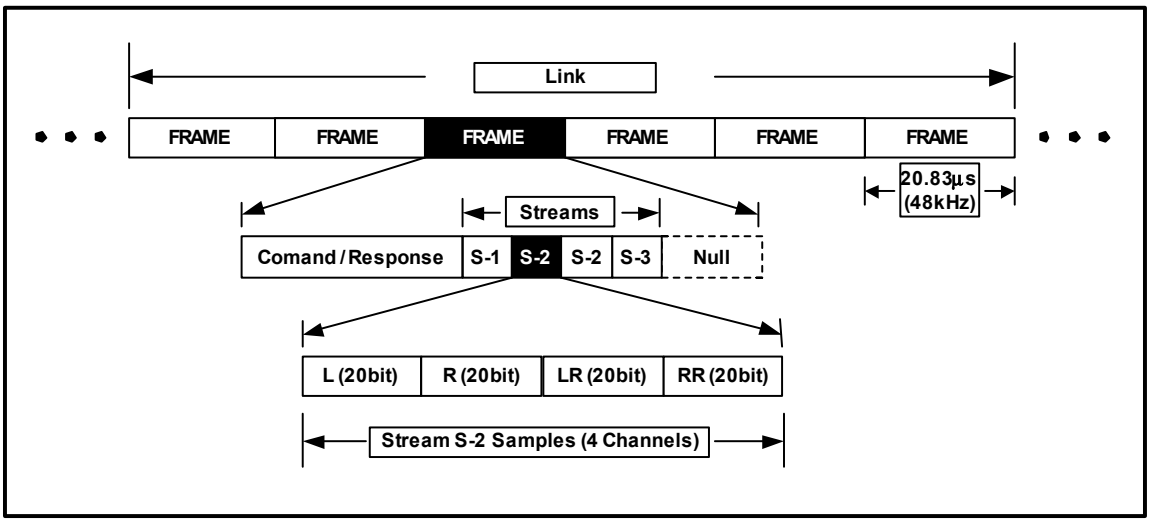

Figure 3. Conceptual Frame Composition

The first breakout shows that each frame contains command or control information and then as many stream sample blocks (labeled S-1, S-2, S-3) as are needed. The total number of streams supportable is limited by the aggregate content of the streams; any unused space in the frame is filled will nulls. Since frames occur at a fixed rate, if a given stream has a sample rate that is higher or lower than 48 kHz, there will be more or less than one sample block in each frame for that stream. Some frames may contain two sample blocks (e.g., S-2 in this illustration) and some may contain none. Section 5.4.1 describes in detail the methods of dealing with sample rates other than 48 kHz.

The second breakout shows that a single stream 2 (S-2) sample block is composed of one sample for each channel in that stream. In this illustration, stream 2 (S-2) has four channels (L, R, LR, RR) and each channel has a 20-bit sample; therefore, the stream sample block uses 80 bits. Note that stream 2 (S-2) is a 96 kHz stream since two sample blocks are transmitted per 20.83 µs (48 kHz) frame.

2.3 DMA Channel Operation

In the High Definition Audio architecture, all audio or modem data is transferred to or from the codecs by the DMA engines in the High Definition Audio controller, which provide standard scatter/gather DMA channel operation. The DMA engines must account for the behavior of their memory attachment port, especially latency, and pre-fetch and buffer enough data on each stream (one or two frames) to ensure there will not be an underrun or overrun on the isochronous High Definition Audio link.

Each DMA engine, when it is enabled, walks through a memory-based list of buffer descriptors, each of which identifies an arbitrary length data buffer. It processes each buffer in turn transferring data to or from the codec. The controller's register space for each DMA engine includes a pointer

to the head of the buffer descriptor list, as well as a register identifying the last valid pointer in the list, as is described in Section 3.1.

2.4 Initialization and Enumeration

Enumeration takes place at three distinct levels in the High Definition Audio architecture.

Controller: The High Definition Audio controller is initialized and enumerated using the standard PCI enumeration mechanisms, or as is appropriate to the system interface. Other than defining memory-mapped register space, that process is outside the scope of this specification.

Codec: After link reset, and during initialization, the link protocol provides each codec on the link a unique ID. This process is described in Section 5.5.3 After the controller has been initialized and the software driver loaded, the software queries each ID on the link to determine the capabilities of the corresponding codec. Hot plugging of codecs is not supported.

Peripheral Devices: When the codec capabilities are enumerated, the types of input/output jacks can be determined through the codec command and control mechanisms under control of the driver. In addition, codec will be able to detect whether a peripheral device is plugged into the jack, and, in some cases, may be able to determine the class of device involved.

3 Register Interface

This chapter outlines the High Definition Audio Controller register interface. All High Definition Audio controllers must implement the registers defined in Section 3.3. The registers defined in Section 3.4 are optional and are not required by this specification.

3.1 Introduction to Controller Registers

3.1.1 Terminology

Table 1 lists terminology used in the definition of registers in this specification.

Table 1. Register Type Definitions

Register Attribute	Meaning
RO	Read only register
RW	Read-Write
RW1C	Read-only status, write-1-to-clear status register
ROS	Sticky bit–Read-only register
RWS	Sticky bit–Read-Write register
RW1CS	Sticky bit–Read-only status, Write-1-to-clear status register.
HWINIT	Hardware initialized; bits are read-only and cannot be reset.
RsvdP	*Reserved*; software must do a read-modify-write to preserve the value of bits.
RsvdZ	*Reserved*; software must use 0's for writes to bits.
RSM	Bit(s) are in the "Resume Well"; i.e., they maintain their state in any power state from which the controller may wake the system.

3.1.2 General Register Behaviors and Access Requirements

All controller registers must be addressable as byte, Word, and Dword quantities. The software must always make register accesses on natural boundaries; Dword accesses must be on Dword boundaries; Word accesses must be on Word boundaries; etc.

Software must also properly handle reserved bits. Reserved bits may be designated "RsvdP" or "RsvdZ." Bits marked "RsvdP" must be preserved using read-modify-writes, while "RsvdZ" bits must be written as 0's. This handling helps to ensure future compatibility.

Note that host controllers are not required to support exclusive-access mechanisms (such as PCI LOCK) for accesses to the memory-mapped register space. Therefore, if software attempts exclusive-access mechanisms to the host controller memory-mapped register space, the results are undefined.

3.1.3 Behavior With 64-bit Addresses

High Definition Audio controllers may have the ability to generate and use 64-bit addresses for system memory. The system software can determine if the controller hardware has this support ability by checking the 64OK bit in the Global Capabilities register. If this bit is a 1, 64-bit addresses can be used.

If the controller does support 64-bit addresses, the system software may use either 64-or 32-bit addresses. All hardware registers which accept a 64 bit address must default the upper 32 bits to 0. If a 32-bit address is then written to the lower 32 bits of the 64-bit register, the address will be correct.

If the controller is capable of generating either 32-or 64-bit addresses, such as a PCI controller, it may be more efficient to generate a 32-bit address if possible, rather than always generating a 64-bit address. In this case, the controller may use a logical "OR" of the upper 32 bits to determine if the address can be correctly represented as a 32-bit address. The specific behavior will depend on the bus on which the High Definition Audio controller resides.

3.2 High Definition Audio Controller System Bus Interface Registers

The High Definition Audio specification does not define the interface-specific registers, such as the PCI configuration space. These registers differ significantly from implementation to implementation, and are well defined in their respective industry specifications.

3.3 High Definition Audio Controller Register Set

Table 2. Controller Registers Summary

Offset Start	Offset End	Symbol	Full Name
00	01	GCAP	Global Capabilities
02	02	VMIN	Minor Version
03	03	VMAJ	Major Version
04	05	OUTPAY	Output Payload Capability
06	07	INPAY	Input Payload Capability
08	0B	GCTL	Global Control
0C	0D	WAKEEN	Wake Enable
0E	0F	WAKESTS	Wake Status
10	11	GSTS	Global Status
12	17	*Rsvd*	*Reserved*
18	19	OUTSTRMPAY	Output Stream Payload Capability
1A	1B	INSTRMPAY	Input Stream Payload Capability
1C	1F	*Rsvd*	*Reserved*
20	23	INTCTL	Interrupt Control

Offset Start	Offset End	Symbol	Full Name
24	27	INTSTS	Interrupt Status
28	2F	*Rsvd*	*Reserved*
30	33	WALCLK	Wall Clock Counter
34	37	*Rsvd*	*Reserved*
38	3B	SSYNC	Stream Synchronization
3C	3F	*Rsvd*	*Reserved*
40	43	CORBLBASE	CORB Lower Base Address
44	47	CORBUBASE	CORB Upper Base Address
48	49	CORBWP	CORB Write Pointer
4A	4B	CORBRP	CORB Read Pointer
4C	4C	CORBCTL	CORB Control
4D	4D	CORBSTS	CORB Status
4E	4E	CORBSIZE	CORB Size
4F	*4F*	*Rsvd*	*Reserved*
50	53	RIRBLBASE	RIRB Lower Base Address
54	57	RIRBUBASE	RIRB Upper Base Address
58	59	RIRBWP	RIRB Write Pointer
5A	5B	RINTCNT	Response Interrupt Count
5C	5C	RIRBCTL	RIRB Control
5D	5D	RIRBSTS	RIRB Status
5E	5E	RIRBSIZE	RIRB Size
5F	*5F*	*Rsvd*	*Reserved*
60	63	ICOI	Immediate Command Output Interface
64	67	ICII	Immediate Command Input Interface
68	69	ICIS	Immediate Command Status
6A	*6F*	*Rsvd*	*Reserved*
70	73	DPIBLBASE	DMA Position Buffer Lower Base
74	77	DPIBUBASE	DMA Position Buffer Upper Base
78	*7F*	*Rsvd*	*Reserved*
80	82	SD0CTL	Input Stream Descriptor 0 Control
83	83	SD0STS	ISD0 Status
84	87	SD0PICB	ISD0 Link Position in Current Buffer
88	8B	SD0CBL	ISD0 Cyclic Buffer Length
8C	8D	SD0LVI	ISD0 Last Valid Index
8E	*8F*	*Rsvd*	*Reserved*
90	91	SD0FIFOD	ISD0 FIFO Size
92	93	SD0FMT	ISD0 Format
94	*97*	*Rsvd*	*Reserved*
98	9B	SD0BDPL	ISD0 Buffer Descriptor List Pointer - Lower
9C	9F	SD0BDPU	ISD0 Buffer Descriptor List Pointer - Upper

Offset Start	Offset End	Symbol	Full Name
A0	*x-1*	*Additional Stream Descriptors*	
*x**	*x*+2	SD*n*CTL	Output Stream Descriptor 0 Control
x+3	*x*+3	SD*n*STS	OSD0 Status
x+4	*x*+7	SD*n*PICB	OSD0 Link Position in Current Buffer
x+8	*x*+B	SD*n*CBL	OSD0 Cyclic Buffer Length
x+C	*x*+D	SD*n*LVI	OSD0 Last Valid Index
x+E	*x+F*	*Rsvd*	*Reserved*
x+10	*x*+11	SD*n*FIFOD	OSD0 FIFO Data
x+12	*x*+13	SD*n*FMT	OSD0 Format
x+14	*x*+17	*Rsvd*	*Reserved*
x+18	x+1B	SD*n*BDPL	OSD0 Buffer Descriptor List Pointer - Lower
x+1C	*x*+1F	SD*n*BDPU	OSD0 Buffer Descriptor List Pointer - Upper
x+20	*y*-1	*Additional Output Stream Descriptors*	
*y**	*y*+2	SD*m*CTL	Bidirectional Stream Descriptor 0 Control
y+3	*y*+3	SD*m*STS	BISD0 Status
y+4	*y*+7	SD*m*PICB	BSD0 Link Position in Current Buffer
y+8	*y*+B	SD*m*CBL	BSD0 Cyclic Buffer Length
y+C	*y*+D	SD*m*LVI	BSD0 Last Valid Index
y+E	*y+F*	*Rsvd*	*Reserved*
y+10	*y*+11	SD*m*FIFOD	BSD0 FIFO Data
y+12	*y*+13	SD*m*FMT	BSD0 Format
y+14	*y*+17	*Rsvd*	*Reserved*
y+18	*y*+1B	SD*m*BDPL	BSD0 Buffer Descriptor List Pointer - Lower
y+1C	*y*+1F	SD*m*BDPU	BSD0 Buffer Descriptor List Pointer - Upper
y+20	*z-1*	*Additional Bidirectional Stream Descriptors*	
2030	2033	WALCLKA	Wall Clock Counter Alias
2084	87	SD0LPIBA	Stream Descriptor 0 Link Position in Current Buffer
20A4	20A7	SD1LPIBA	Stream Descriptor 1 Link Position in Current Buffer
…	…	…	*Other Link Position in Current Buffer registers*

** The offsets indicated by "x", "y", and "z" are dependent on the number of Input and Output Stream Descriptors as indicated by the ISS and OSS values in the Global Capabilities Register (see Section 3.3.1).*

*x = 80h + (ISS * 20h)*

*y = 80h + (ISS * 20h) + (OSS * 20h)*

*z = 80h + (ISS * 20h) + (OSS * 20h) + (BSS * 20h)*

3.3.1 Global Capabilities, Status, and Control

The Global Capabilities register indicates the capabilities of the controller.

The Global Control register provides global level control of the controller and link. This includes controller and link power management.

3.3.2 Offset 00h: GCAP – Global Capabilities

Length: 2 bytes

Table 3. Global Capabilities

Bit	Type	Description
15:12	RO	**Number of Output Streams Supported (OSS):** A value of 0000b indicates that there are no Output Streams supported. A value of maximum 15 output streams are supported. 0000b: No output streams supported 0001b: 1 output stream supported … 1111b: 15 output streams supported
11:8	RO	**Number of Input Streams Supported (ISS):** A value of 0000b indicates that there are no Input Streams supported. A maximum of 15 input streams are supported. 0000b: No input streams supported 0001b: 1 input stream supported … 1111b: 15 input streams supported
7:3	RO	**Number of Bidirectional Streams Supported (BSS):** A value of 00000b indicates that there are no Bidirectional Streams supported. A maximum of 30 bidirectional streams are supported. 00000b: No bidirectional streams supported 00001b: 1 bidirectional stream supported … 11110b: 30 bidirectional streams supported
2:1	RO	**Number of Serial Data Out Signals (NSDO):** A 0 indicates that one SDO line is supported; a 1 indicates that two SDO lines are supported. Software can enable the use of striping by setting the appropriate bit in the Stream Buffer Descriptor. 00: 1 SDO 01: 2 SDOs 10: 4 SDOs 11: *Reserved*
0	RO	**64 Bit Address Supported (64OK):** A 1 indicates that 64 bit addressing is supported by the controller for BDL addresses, data buffer addresses, and command buffer addresses. A 0 indicates that only 32-bit addressing is available.

There are a maximum of 30 Streams that can be supported, of which 15 may be configured as output and 15 may be configured as input streams at any one point in time.

3.3.3 Offset 02h: VMIN – Minor Version

Length: 1 byte

Table 4. Minor Version

Bit	Type	Reset	Description
7:0	RO	00h	**Minor Version (VMIN):** Indicates minor revision number 00h of the High Definition Audio specification, for specification version "1.0."

3.3.4 Offset 03h: VMAJ – Major Version

Length: 1 byte

Table 5. Major Version

Bit	Type	Reset	Description
7:0	RO	01h	**Major Version (VMAJ):** indicates major revision number 1 of the High Definition Audio specification, for specification version "1.0."

3.3.5 Offset 04h: OUTPAY – Output Payload Capability

Length: 2 bytes

Table 6. Output Payload Capability

Bit	Type	Reset	Description
15:0	RO	3Ch	**Output Payload Capability (OUTPAY):** Indicates the total output payload available on the link. This does not include bandwidth used for command and control. This measurement is in 16-bit Word quantities per 48-kHz frame. The default link clock speed of 24.000 MHz (the data is double pumped) provides 1000 bits per frame, or 62.5 Words in total. Forty bits (2.5 Words) are used for command and control, leaving 60 Words available for data payload. Note that this value does not reflect any bandwidth increase due to support for multiple SDO lines. 00h: 0 Words 01h: 1 Word payload … FFh: 255h Word payload

3.3.6 Offset 06h: INPAY – Input Payload Capability

Length: 2 bytes

Table 7. Input Payload Capability

Bit	Type	Reset	Description
15:0	RO	1Dh	**Input Payload Capability (INPAY):** Indicates the total input payload available on the link. This does not include bandwidth used for command and control. This measurement is in 16-bit Word quantities per 48-kHz frame. The default link clock speed of 24.000 MHz provides 500 bits per frame, or 31.25 Words. 36 bits (2.25 Words) are used for command and control, leaving 29 Words for payload. This measurement is on a per-codec basis. 00h: 0 Words 01h: 1 Word payload … FFh: 255h Word payload

3.3.7 Offset 08h: GCTL – Global Control

Length: 4 bytes

Table 8. Global Control Register

Bit	Type	Reset	Description
31:9	RsvP	0's	*Reserved*
8	RW	0	**Accept Unsolicited Response Enable (UNSOL):** If UNSOL is a 1, Unsolicited Responses from the codecs are accepted by the controller and placed into the Response Input Ring Buffer. If UNSOL is a 0, unsolicited responses are not accepted and dropped on the floor.
7:2	RsvP	0's	*Reserved*
1	RW	0	**Flush Control (FCNTRL):** Writing a 1 to this bit initiates a flush. The flush is complete when Flush Status is set. Before a flush cycle is initiated, the DMA Position Buffer must be programmed with a valid memory address by software, but the DMA Position Buffer bit 0 need not be set to enable the position reporting mechanism. Also, all streams must be stopped (the associated RUN bit must be 0). When the flush is initiated, the controller will flush pipelines to memory to ensure that the hardware is ready to transition to a D3 state. Setting this bit is not a critical step in the power state transition if the content of the FIFOs is not critical.

Bit	Type	Reset	Description
0	RWS	HwInit	**Controller Reset (CRST):** Writing a 0 to this bit causes the High Definition Audio controller to transition to the Reset state. With the exception of certain registers such as those required for Wake support, all state machines, FIFO's, and memory mapped configuration registers (not PCI Configuration Registers) in the controller will be reset. The link RESET# signal will be asserted and all other link signals will be driven to their "reset" values. After the hardware has completed sequencing into the reset state, it will report a 0 in this bit. Software must read a 0 from this bit to verify that the controller is in reset. Writing a 1 to this bit causes the controller to exit its Reset state and deassert the link RESET# signal. Software is responsible for setting/clearing this bit such that the minimum link RESET# signal assertion pulse width specification is met (see Section 5.5). When the controller hardware is ready to begin operation, it will report a 1 in this bit. Software must read a 1 from this bit before accessing any controller registers. The CRST# bit defaults to a 0 after hardware reset, therefore software needs to write a 1 to this bit to begin operation. Note that the CORB/RIRB RUN bits and all Stream RUN bits must be verified cleared to 0 before CRST# is written to 0 (asserted) in order to assure a clean re-start. When CRST is 0 indicating that the controller is in reset, most registers will return their default values on reads, and writes will have no effect. The exceptions are the WAKEEN and STATESTS registers, which are only cleared on power-on reset, and the CRST bit itself, which will cause the controller to leave the reset state when a 1 is written to it.

3.3.8 Offset 0Ch: WAKEEN – Wake Enable

Length: 2 bytes

Table 9. Wake Enable

Bit	Type	Reset	Description
15	RsvP	0	*Reserved*
14:0	RW, RSM	0	**SDIN Wake Enable Flags (SDIWEN):** Bits that control which SDIN signal(s) may generate a wake event or processor interrupt in response to a codec State Change request. A 1 bit in the bit mask indicates that the associated SDIN signal is enabled to generate a wake or processor interrupt. The SDATA_IN[0] signal corresponds to bit 0, etc. These bits are only cleared by a power-on reset. Software must make no assumptions about how these bits are set and set them appropriately.

The WAKEEN bits are used to indicate which bits in the STATESTS register may cause either a wake event or an interrupt. Depending on the system implementation and the current system power state and policy, the controller should generate either a processor interrupt or a system wake signal, as appropriate, to signal the status change.

3.3.9 Offset 0Eh: STATESTS – State Change Status

Length: 2 bytes

Table 10. Wake Status

Bit	Type	Reset	Description
15	RsvdZ	0's	*Reserved*
14:0	RW1CS, RSM	0's	**SDIN State Change Status Flags (SDIWAKE):** Flag bits that indicate which SDIN signal(s) received a "State Change" event. The bits are cleared by writing 1's to them. The SDATA_IN[0] line corresponds to bit 0, etc. These bits are only cleared by a power-on reset.

The SDIWAKE bits are used to indicate that a "Status Change" event has occurred on the link, which usually indicates that either the codec has just come out of reset and is requesting an address, or that a codec is signaling a wake event. The controller hardware will perform the resulting address cycle on the bus, and set this bit to inform the software that the event has occurred. The setting of this bit may cause a processor interrupt to occur if the appropriate WAKEEN bits (Section 3.3.8) are set.

3.3.10 Offset 10h: GSTS – Global Status

Length: 2 bytes

Table 11. Global Status

Bit	Type	Reset	Description
15:2	RsvdZ	0's	*Reserved*
1	RW1C	0	**Flush Status (FSTS):** This bit is set to a 1 by the hardware to indicate that the flush cycle initiated when the FCNTRL bit was set has completed. Software must write a 1 to clear this bit before the next time FCNTRL is set to clear the bit.
0	RsvdZ	0	*Reserved*

3.3.11 Offset 18h: OUTSTRMPAY – Output Stream Payload Capability

Length: 2 bytes

Table 12. Output Payload Capability

Bit	Type	Reset	Description
15:0	RO	3Ch	**Output Stream Payload Capability (OUTSTRMPAY):** Indicates the maximum number of Words per frame for any single output stream. This measurement is in 16-bit Word quantities per 48-kHz frame. The value must not be larger than the OUTPAY register value. Software must ensure that a format which would cause more Words per frame than indicated is not programmed into the Output Stream Descriptor Register. 00h: No Limit (Stream size is limited only by OUTPAY) 01h: 1 Word payload … FFh: 255h Word payload

3.3.12 Offset 1Ah: INSTRMPAY – Input Stream Payload Capability

Length: 2 bytes

Table 13. Input Payload Capability

Bit	Type	Reset	Description
15:0	RO	1Dh	**Input Stream Payload Capability (INSTRMPAY)** Indicates the maximum number of Words per frame for any single input stream. This measurement is in 16-bit Word quantities per 48-kHz frame. The value must not be larger than the INPAY register value. Software must ensure that a format which would cause more Words per frame than indicated is not programmed into the Input Stream Descriptor Register. 00h: No Limit (Stream size is limited only by INPAY) 01h: 1 Word payload … FFh: 255h Word payload

3.3.13 Interrupt Status and Control

The Interrupt Status and Control register provides a central point for controlling and monitoring interrupt generation. The SIE (Stream Interrupt Enable) register controls the interrupt mask for each individual Input, Output, or Bidirectional Stream. Setting a 1 in the appropriate bit allows the particular interrupt source to generate a processor interrupt.

The SIS (Stream Interrupt Status) register indicates the current interrupt status of each interrupt source. A 1 indicates that an interrupt is being requested. Note that the state of these bits is independent of the SIE bits; even if the corresponding bit is set to a 0 in the Stream Interrupt Enable register to disable processor interrupt generation, the Status bit may still be set to indicate that stream is requesting service. This can be used by polling software to determine which Streams need attention without incurring system interrupts.

The CIE (Controller Interrupt Enable) and CIS (Controller Interrupt Status) control and indicate the status of the general controller interrupt. General controller interrupt sources are Response Input Ring Buffer activity and Wake (Status Change) interrupts from codecs.

The GIE (Global Interrupt Enable) and GIS (Global Interrupt Status) control and indicate the status of all hardware interrupt sources in the High Definition Audio controller. If the GIS bit is 1, the controller needs CPU servicing. If GIE is a 1, a processor interrupt will be requested when GIS is 1; if GIE is a 0, then no processor interrupt will be requested, although GIS will still be set indicating that servicing is desired.

3.3.14 Offset 20h: INTCTL – Interrupt Control

Length: 4 bytes

Table 14. Interrupt Control Register

Bit	Type	Reset	Description
31	RW	0	**Global Interrupt Enable (GIE):** Global bit to enable device interrupt generation. When set to 1, the High Definition Audio device is enabled to generate an interrupt. This control is in addition to any bits in the bus specific address space, such as the Interrupt Enable bit in the PCI Configuration Space.
30	RW	0	**Controller Interrupt Enable (CIE):** Enables the general interrupt for controller functions. When set to 1 the controller generates an interrupt when the corresponding status bit get sets due to a Response Interrupt, a Response Buffer Overrun, and wake events.
29:0	RW	0h	**Stream Interrupt Enable (SIE):** When set to 1, the individual Streams are enabled to generate an interrupt when the corresponding stream status bits get set. A stream interrupt will be caused as a result of a buffer with IOC = 1 in the BDL entry being completed or as a result of a FIFO error (underrun or overrun) occurring. Control over the generation of each of these sources is in the associated Stream Descriptor. The streams are numbered and the SIE bits assigned sequentially, based on their order in the register set. For instance, if there are two input streams, three output streams, and one bidirectional stream (ISS = 2, OSS = 3, BSS = 1), the bit assignments would be as follows: Bit 0: Input Stream 1 Bit 1: Input Stream 2 Bit 2: Output Stream 1 Bit 3: Output Stream 2 Bit 4: Output Stream 3 Bit 5: Bidirectional Stream 1 Bits 6-28: *Reserved*

3.3.15 Offset 24h: INTSTS – Interrupt Status

Length: 4 bytes

Table 15. Interrupt Status Register

Bit	Type	Reset	Description
31	RO	0	**Global Interrupt Status (GIS):** This bit is an "OR" of all of the interrupt status bits in this register.
30	RW1C	0	**Controller Interrupt Status (CIS):** Status of general controller interrupt. This bit may be set regardless of the corresponding enable bit, but a hardware interrupt will not be generated unless the corresponding enable bit is set. A 1 indicates that an interrupt condition occurred due to a Response Interrupt, a Response Overrun, or a Codec State Change request. The exact cause can be determined by interrogating the RIRB Status register and the State Change Status register. Note that this bit is set regardless of the state of the corresponding interrupt enable bit. This bit is cleared by writing a 1.
29:0	RW1C	0h	**Stream Interrupt Status (SIS):** A 1 indicates that an interrupt condition occurred on the corresponding Stream. Note that these status bits are set regardless of the state of the corresponding interrupt enable bits. The bits are cleared by writing 1's to them. The streams are numbered and the SIS bits assigned sequentially based on their order in the register set in the same way the SIE bits are set. (Section 3.3.14).

3.3.16 Offset 30h: Wall Clock Counter

The 32-bit monotonic counter provides a "wall clock" that can be used by system software to synchronize independent audio controllers. The counter must be implemented.

Length: 4 bytes

Table 16. Wall Clock Counter

Bit	Type	Reset	Description
31:0	RO	0000_0000h	**Wall Clock Counter (Counter):** 32 bit counter that is incremented at the link bitclock rate and rolls over from FFFF_FFFFh to 0000_0000h. This counter will roll over to 0 with a period of approximately 179 seconds with the nominal 24-MHz bitclock rate. This counter is enabled while the BCLK bit is set to 1. Software uses this counter to synchronize between multiple controllers. The counter will be reset on controller reset.

3.3.17 Offset 38h: SSYNC – Stream Synchronization

The Stream Synchronization bits provide a mechanism for synchronously starting or stopping two or more streams so that the streams have a common time reference.

Length: 4 bytes

Table 17. Stream Synchronization

Bit	Type	Reset	Description
31:30	RsvP	0's	*Reserved*
29:0	RW	0's	**Stream Synchronization Bits (SSYNC):** The Stream Synchronization bits, when set, block data from being sent on or received from the link. Each bit controls the associated Stream Descriptor; bit 0 corresponds to the first Stream Descriptor, etc. To synchronously start a set of DMA engines, the bits in the SSYNC register are set to a 1. The RUN bits for the associated Stream Descriptors can be set to a 1 to start the DMA engines. When all streams are ready, the associated SSYNC bits can all be set to 0 at the same time, and transmission or reception of bits to or from the link will begin together at the start of the next full link frame. To synchronously stop streams, the bits are set in the SSYNC register, and the RUN bits in the Stream Descriptors are cleared by software.

3.3.18 Offset 40h: CORB Lower Base Address

Length: 4 bytes

Table 18. CORB Lower Base Address

Bit	Type	Reset	Description
31:7	RW	0's	**CORB Lower Base Address (CORBLBASE):** Lower address of the Command Output Ring Buffer, allowing the CORB Base Address to be assigned on any 1 KB boundary. This register field must not be written when the DMA engine is running or the DMA transfer may be corrupted.
6:0	RO	0's	**CORB Lower Base Unimplemented Bits:** Hardwired to 0. This requires the CORB to be allocated with 128-byte granularity to allow for cache line fetch optimizations.

3.3.19 Offset 44h: CORBUBASE – CORB Upper Base Address

Length: 4 bytes

Table 19. CORB Upper Base Address

Bit	Type	Reset	Description
31:0	RW	0000_0000h	**CORB Upper Base Address (CORBUBASE):** Upper 32 bits of address of the Command Output Ring Buffer. This register field must not be written when the DMA engine is running or the DMA transfer may be corrupted. This register is reserved, read only 0 if the 64OK bit indicates that the controller does not support 64-bit addressing.

3.3.20 Offset 48h: CORBWP – CORB Write Pointer

Length: 2 bytes

Table 20. CORB Write Pointer

Bit	Type	Reset	Description
15:8	RsvP	0's	*Reserved*
7:0	RW	0's	**CORB Write Pointer (CORBWP):** Software writes the last valid CORB entry offset into this field in Dword granularity. The DMA engine fetches commands from the CORB until the Read Pointer matches the Write Pointer. This supports up to 256 CORB entries (256 x 4 B = 1 KB). This field may be written while the DMA engine is running.

3.3.21 Offset 4Ah: CORBRP – CORB Read Pointer

Length: 2 bytes

Table 21. CORB Read Pointer

Bit	Type	Reset	Description
15	W	0	**CORB Read Pointer Reset (CORBRPRST):** Software writes a 1 to this bit to reset the CORB Read Pointer to 0. The DMA engine must be stopped prior to resetting the Read Pointer or else DMA transfer may be corrupted. This bit will always be read as 0.
14:8	RsvP	0's	*Reserved*
7:0	R0	0's	**CORB Read Pointer (CORBRP):** Software reads this field to determine how many commands it can write to the CORB without over-running. The value read indicates the CORB Read Pointer offset in Dword granularity. The offset entry read from this field has been successfully fetched by the DMA controller and may be over-written by software. Supports up to 256 CORB entries (256 x 4 B = 1 KB) in the cyclic CORB buffer. This field may be read while the DMA engine is running.

3.3.22 Offset 4Ch: CORBCTL – CORB Control

Length: 1 byte

Table 22. CORB Control

Bit	Type	Reset	Description
7:2	RsvdP	0's	*Reserved*
1	RW	0	**Enable CORB DMA Engine (CORBRUN):** 0 = DMA Stop 1 = DMA Run (when Read Pointer lags Write Pointer) Must read the value back
0	RW	0	**CORB Memory Error Interrupt Enable (CMEIE):** If this bit is set, the controller will generate and interrupt if the MEI status bit is set.

3.3.23 Offset 4Dh: CORBSTS – CORB Status

Length: 1 byte

Table 23. CORB Status

Bit	Type	Reset	Description
7:1	RsvdZ	0's	*Reserved*
0	RW1C	0	**CORB Memory Error Indication (CMEI):** If this status bit is set, the controller has detected an error in the pathway between the controller and memory. This may be an ECC bit error or any other type of detectable data error which renders the command data fetched invalid. Writing a 1 to this bit will clear the bit, but a CRST must be performed before operation continues as this indicates a severe machine error has occurred and the current state is not trustable.

3.3.24 Offset 4Eh: CORBSIZE – CORB Size

Length: 1 byte

Table 24. CORB Size

Bit	Type	Reset	Description
7:4	RO	Imp.Dep	**CORB Size Capability (CORBSZCAP):** A bit mask indicating the sizes of the CORB supported by the controller. Bits [7:4] / CORB Size 0001 / 8 B = 2 entries 0010 / 64 B = 16 entries 0100 / 1024 B = 256 Entries 1000 / *Reserved* This is implemented as a bit mask; for example, if the controller supported two entries and 256 entries, this register would have a value of 0101b.
3:2	RsvdP	0	*Reserved*
1:0	RW	Imp.Dep	**CORB Size (CORBSIZE):** The setting of the register determines when the address counter in the DMA controller will wrap around. Bits [1:0] / CORB Size 00 / 8 B = 2 entries 01 / 64 B = 16 entries 10 / 1 KB = 256 entries 11 / *Reserved*

3.3.25 Offset 50h: RIRBLBASE – RIRB Lower Base Address

Length: 4 bytes

Table 25. RIRB Lower Base Address

Bit	Type	Reset	Description
31:7	RW	0's	**RIRB Lower Base Address (RIRBLBASE):** Lower address of the Response Input Ring Buffer, allowing the RIRB Base Address to be assigned on any 2-KB boundary. This register field must not be written when the DMA engine is running or the DMA transfer may be corrupted.
6:0	RO	0's	**RIRB Lower Base Unimplemented Bits:** Hardwired to 0 to force 128-byte buffer alignment for cache line fetch optimizations.

3.3.26 Offset 54h: RIRBUBASE – RIRB Upper Base Address

Length: 4 bytes

Table 26. RIRB Upper Base Address

Bit	Type	Reset	Description
31:0	RW	0000_0000h	**RIRB Upper Base Address (RIRBUBASE):** Upper 32 bits of address of the Response Input Ring Buffer. This register field must not be written when the DMA engine is running or the DMA transfer may be corrupted. This register is reserved, read only 0 if the 64OK bit indicates that the controller does not support 64-bit addressing.

3.3.27 Offset 58h: RIRBWP – RIRB Write Pointer

Length: 2 bytes

Table 27. RIRB Write Pointer

Bit	Type	Reset	Description
15	W	0	**RIRB Write Pointer Reset (RIRBWPRST):** Software writes a 1 to this bit to reset the RIRB Write Pointer and to 0's. The DMA engine must be stopped prior to resetting the Write Pointer or else DMA transfer may be corrupted. This bit will always be read as 0.
14:8	RsvP	0's	*Reserved*
7:0	RO	0's	**RIRB Write Pointer (RIRBWP):** Indicates the last valid RIRB entry written by the DMA controller. Software reads this field to determine how many responses it can read from the RIRB. The value read indicates the RIRB Write Pointer offset in two Dword units (since each RIRB entry is two Dwords long). Supports up to 256 RIRB entries (256 x 8 B = 2 KB) in the cyclic RIRB buffer. This field may be read while the DMA engine is running.

3.3.28 Offset 5Ah: RINTCNT – Response Interrupt Count

Length: 2 bytes

Table 28. RIRB Response Interrupt Count

Bit	Type	Reset	Description
15:8	RsvP	0's	*Reserved*
7:0	RW	00h	**N Response Interrupt Count (RINTCNT):** 0000_0001b = 1 Response sent to RIRB … 1111_1111b = 255 Responses sent to RIRB 0000_0000b = 256 Responses sent to RIRB The DMA engine should be stopped when changing this field or else an interrupt may be lost. Note that each Response occupies two Dwords in the RIRB. This is compared to the total number of responses that have been returned, as opposed to the number of frames in which there were responses. If more than one codec responds in one frame, then the count is increased by the number of responses received in the frame.

3.3.29 Offset 5Ch: RIRBCTL – RIRB Control

Length: 1 byte

Table 29. RIRB Control

Bit	Type	Reset	Description
7:3	RsvP	0's	*Reserved*
2	RW	0	**Response Overrun Interrupt Control (RIRBOIC):** If this bit is set, the hardware will generate an interrupt when the Response Overrun Interrupt Status bit is set.
1	RW	0	**RIRB DMA Enable (RIRBDMAEN):** 0 = DMA Stop 1 = DMA Run (when Response queue not empty)
0	RW	0	**Response Interrupt Control (RINTCTL):** 0 = Disable Interrupt 1 = Generate an interrupt after N number of Responses are sent to the RIRB buffer or when an empty Response slot is encountered on all SDATA_IN_x inputs after a frame which returned a response (whichever occurs first). The N counter is reset when the interrupt is generated.

3.3.30 Offset 5Dh: RIRBSTS – RIRB Status

Length: 1 bytes

Table 30. RIRB Status

Bit	Type	Reset	Description
7:3	RsvdZ	0's	*Reserved*
2	RW1C	0	**Response Overrun Interrupt Status (RIRBOIS):** Hardware sets this bit to a 1 when an overrun occurs in the RIRB. An interrupt may be generated if the Response Overrun Interrupt Control bit is set. This bit will be set if the RIRB DMA engine is not able to write the incoming responses to memory before additional incoming responses overrun the internal FIFO. When hardware detects an overrun, it will drop the responses which overrun the buffer and set the RIRBOIS status bit to indicate the error condition. Optionally, if the RIRBOIC is set, the hardware will also generate an error to alert software to the problem. Software clears this bit by writing a 1 to it.
1	RsvdZ	0's	*Reserved*
0	RW1C	0	**Response Interrupt (RINTFL):** Hardware sets this bit to a 1 when an interrupt has been generated after N number of Responses are sent to the RIRB buffer or when an empty Response slot is encountered on all SDATA_IN[x] inputs (whichever occurs first). Software clears this flag by writing a 1 to this bit.

3.3.31 Offset 5Eh: RIRBSIZE – RIRB Size

Length: 1 byte

Table 31. RIRB Size

Bit	Type	Reset	Description
7:4	RO	Imp.Dep	**RIRB Size Capability (RIRBSZCAP):** A bit mask identifying the possible sizes of the RIRB. Bits [7:4] / RIRB Size 0001 / 16 B = 2 entries 0010 / 128 B = 16 entries 0100 / 2048 B = 256 Entries 1000 / *Reserved* This implemented as a bit mask; for example, if the controller supported two entries and 256 entries, this register would be Read Only 0101b.
3:2	RsvdP	0	*Reserved*

Bit	Type	Reset	Description
1:0	RW	Imp.Dep	**RIRB Size (RIRBSIZE):** The setting of the register determines when the address counter in the DMA controller will wrap around. Bits [1:0] / RIRB Size 00 / 16 B = 2 entries 01 / 128 B = 16 entries 10 / 2 KB = 256 entries 11 / *Reserved* This value must not be changed when the RIRB DMA engine is enabled.

3.3.32 Offset 70h: DPLBASE – DMA Position Lower Base Address

Length: 4 bytes

Table 32. DMA Position Lower Base Address

Bit	Type	Reset	Description
31:7	RW	0's	**DMA Position Lower Base Address (DPLBASE):** Lower 32 bits of the DMA Position Buffer Base Address. This register field must not be written when any DMA engine is running or the DMA transfer may be corrupted. This same address is used by the Flush Control, and must be programmed with a valid value before the FLCNRTL bit is set.
6:1	RO	0's	**DMA Position Lower Base Unimplemented Bits:** Hardwired to 0 to force 128-byte buffer alignment for cache line write optimizations.
0	RW	0	**DMA Position Buffer Enable:** When this bit is set to a 1, the controller will write the DMA positions of each of the DMA engines to the buffer in main memory periodically. Software can use this value to know what data in memory is valid data. The controller must ensure that the values in the DMA Position Buffer that the software can read represent positions in the stream for which valid data exists in the Stream's DMA buffer. This has particular relevance in systems which support isochronous transfer; the stream positions in the software-visible memory buffer must represent stream data which has reached the Global Observation point.

3.3.33 Offset 74h: DPUBASE – DMA Position Upper Base Address

Length: 4 bytes

Table 33. DMA Position Upper Base Address

Bit	Type	Reset	Description
31:0	RW	0000_0000h	**DMA Position Upper Base Address (RIRBUBASE):** Upper 32 bits of address of the DMA Position Buffer Base Address. This register field must not be written when the DMA engine is running or the DMA transfer may be corrupted. This register is reserved, read only 0 if the 64OK bit indicates that the controller does not support 64-bit addressing.

3.3.34 Stream Descriptors

The Stream description registers control the DMA engines which transfer the payload data to and from the High Definition Audio link. The Input, Output, and Bidirectional descriptors share the same definition, with minor changes in the definitions of some bits to accommodate the slightly different behavior of the engines.

3.3.35 Offset 80: {IOB}SDnCTL – Input/Output/Bidirectional Stream Descriptor *n* Control

Length: 3 bytes

Table 34. Stream Descriptor *n* Control

Bit	Type	Reset	Description
23:20	RW	0h	**Stream Number (STRM):** This value reflects the Tag associated with the data being transferred on the link. When data controlled by this descriptor is sent out over the link, it will have this stream number encoded on the SYNC signal. When an input stream is detected on any of the SDATA_INx signals that match this value, the data samples are loaded into the FIFO associated with this descriptor. Note that while a single SDAT_INx input may contain data from more than one stream number, two different SDATA_INx inputs may not be configured with the same stream number. Although the controller hardware is capable of transmitting any stream number, by convention stream 0 is reserved as unused by software, so that converters whose stream numbers have been reset to 0 do not unintentionally decode data not intended for them. 0000 = *Reserved* (Indicates Unused) 0001 = Stream 1 … 1110 = Stream 14 1111 = Stream 15
19	RW	0's	**Bidirectional Direction Control (DIR):** (Bidirectional engines only. Read-only 0 for engines which are not bidirectional.) For a bidirectional engine, this bit determines the direction in which the bidirectional engine should operate. This bit can only be changed after stream reset (SRST) has been asserted and cleared and before any other stream registers have been programmed. Because setting this bit changes the fundamental behavior of the stream and the meaning of some bits, changing this bit after any other register in the stream descriptor has been written to may lead to undetermined results. 0 = Bidirectional engine is configured as an Input Engine. 1 = Bidirectional engine is configured as an Output Engine.

Bit	Type	Reset	Description
18	RW	0h	**Traffic Priority (TP):** If set to a 1, the stream will be treated as preferred traffic if the underlying bus supports it. If set to a 0, the traffic will be handled on a "best effort" basis. The actual meaning of this bit is specific to the hardware implementation. Depending on the hardware implementation, there may be additional restrictions on the traffic, and software should assume that the buffers associated with this stream will not be snooped or cached. On PCI Express*, for example, setting the TP bit to a 1 might cause the controller to generate non-snooped isochronous traffic, while a PCI implementation may ignore this bit.
17:16	RW	00	**Stripe Control (STRIPE):** (Output and Bidirectional engines configured for output only. Read Only 0 for input streams.) If the NSDO field of the Global Capabilities register indicates that the controller supports multiple SDO lines and the codec has been determined to have compatible capabilities, STRIPE can be used to indicate how many of the SDO lines the stream should be striped across. 00: 1 SDO 01: 2 SDOs 10: 4 SDOs 11: *Reserved*
15:5	RsvP	0's	*Reserved*
4	RW	0	**Descriptor Error Interrupt Enable (DEIE):** Controls whether an interrupt is generated when the Descriptor Error Status (DESE) bit is set.
3	RW	0's	**FIFO Error Interrupt Enable (FEIE):** This bit controls whether the occurrence of a FIFO error (overrun for input or under run for output) will cause an interrupt or not. If this bit is not set, bit 4 in the Status register will be set, but the interrupt will not occur. Either way, the samples will be dropped.
2	RW	0	**Interrupt On Completion Enable (IOCE):** This bit controls whether or not an interrupt occurs when a buffer completes with the IOC bit set in its descriptor. If this bit is not set, bit 3 in the Status register will be set, but the interrupt will not occur.
1	RW	0	**Stream Run (RUN):** When set to 1, the DMA engine associated with this input stream will be enabled to transfer data in the FIFO to main memory. When cleared to 0, the DMA engine associated with this input stream will be disabled. If the corresponding SSYNC bit is 0, input stream data will be taken from the link and moved to the FIFO and an over-run may occur.
0	RW	0	**Stream Reset (SRST):** Writing a 1 causes the corresponding stream to be reset. The Stream Descriptor registers (except the SRST bit itself), FIFO's, and cadence generator for the corresponding stream are reset. After the stream hardware has completed sequencing into the reset state, it will report a 1 in this bit. Software must read a 1 from this bit to verify that the stream is in reset. Writing a 0 causes the corresponding stream to exit reset. When the stream hardware is ready to begin operation, it will report a 0 in this bit. Software must read a 0 from this bit before accessing any of the stream registers. The RUN bit must be cleared before SRST is asserted.

3.3.36 Offset 83h: {IOB}SD0STS – Input/Output/Bidirectional Stream Descriptor *n* Status

Length: 1 byte

Table 35. Stream Descriptor *n* Status

Bit	Type	Reset	Description
7:6	RsvdZ	0	*Reserved*.
5	RO	0	**FIFO Ready (FIFORDY):** For an Output stream, the controller hardware will set this bit to a 1 while the output DMA FIFO contains enough data to maintain the stream on the link. This bit defaults to 0 on reset because the FIFO is cleared on a reset. The amount of data required to maintain the stream will depend on the controller implementation but, in general, for an output stream, it means that the FIFO is full. For an input stream, this bit indicates that a descriptor has been fetched, and the engine is ready for the RUN bit to be set.
4	RW1C	0	**Descriptor Error (DESE):** During the fetch of a descriptor, an error has occurred. This could be a result of a Master Abort, a Parity or ECC error on the bus, or any other error that renders the current Buffer Descriptor or Buffer Descriptor List useless. This error is treated as a fatal stream error as the stream cannot continue running. The RUN bit will be cleared and the stream will stop. Software may attempt to restart the stream engine after addressing the cause of the error and writing a 1 to this bit to clear it.
3	RW1C	0	**FIFO Error (FIFOE):** Set when a FIFO error occurs. Bit is cleared by writing a 1 to this bit position. This bit is set even if an interrupt is not enabled. For an input stream, this indicates a FIFO overrun occurring while the RUN bit is set. When this happens, the FIFO pointers do not increment and the incoming data is not written into the FIFO, thereby being lost. For an output stream, this indicates a FIFO under run when there are still buffers to send. The hardware should not transmit anything on the link for the associated stream if there is not valid data to send.
2	RW1C	0	**Buffer Completion Interrupt Status (BCIS):** For an outbound engine, this bit is set to 1 by the hardware after the last byte of data for the current descriptor has been fetched from memory and put into the DMA FIFO, and the current descriptor has the IOC bit set. For an inbound engine, this bit is set to 1 by the hardware after the last byte of data for the current descriptor with an IOC bit set has been removed from the DMA FIFO and the current descriptor has the IOC bit set. BCIS remains active until software clears it by writing a 1 to this bit position.
1:0	RsvdZ	0	*Reserved*.

3.3.37 Offset 84: {IOB}SDnLPIB – Input/Output/Bidirectional Stream Descriptor *n* Link Position in Buffer

Length: 4 bytes

Table 36. Stream Descriptor *n* Link Position in Buffer

Bit	Type	Reset	Description
31:0	RO	0's	**Link Position in Buffer (LPIB):** Indicates the number of bytes that have been received off the link. Since this register reflects the number of bytes that have been received into the current buffer, for the first buffer SDnLPIB will count from 0 to the value in the Cyclic Buffer Length (SDnCBL) register, inclusive. For subsequent buffers, SNnLPIB will count from a value of 1 to the value in the Cyclic Buffer Length register, inclusive.

3.3.38 Offset 88: {IOB}SDnCBL – Input/Output/Bidirectional Stream Descriptor *n* Cyclic Buffer Length

Length: 4 bytes

Table 37. Stream Descriptor *n* Cyclic Buffer Length

Bit	Type	Reset	Description
31:0	RW	0's	**Cyclic Buffer Length (CBL):** Indicates the number of bytes in the complete cyclic buffer. Link Position in Buffer (SDnLPIB) will be reset when it reaches this value. Software may only write to this register after Global Reset, Controller Reset, or Stream Reset has occurred. Once the RUN bit has been set to enable the engine, software must not write to this register until after the next reset is asserted, or undefined events will occur. CBL must represent an integer number of samples. This value should not be modified except when the RUN bit is 0.

3.3.39 Offset 8C: {IOB}ISDnLVI – Input/Output/Bidirectional Stream Descriptor *n* Last Valid Index

Length: 2 bytes

Table 38. Stream Descriptor *n* Last Valid Index

Bit	Type	Reset	Description
15:8	RsvP	0's	*Reserved*
7:0	RW	00h	**Last Valid Index (LVI):** The value written to this register indicates the index for the last valid Buffer Descriptor in the BDL. After the controller has processed this descriptor, it will wrap back to the first descriptor in the list on continue processing. LVI must be at least 1; i.e., there must be at least two valid entries in the buffer descriptor list before DMA operations can begin. This value should not be modified except when the RUN bit is 0.

3.3.40 Offset 90: {IOB}SDnFIFOS – Input/Output/Bidirectional Stream Descriptor *n* FIFO Size

Length: 2 bytes

Table 39. Stream Descriptor *n* FIFO Size

Bit	Type	Reset	Description
15:0	RO	0000h	**FIFO Size (FIFOS):** Indicates the maximum number of bytes that could be fetched by the controller at one time. This is the maximum number of bytes that may have been DMA'd into memory but not yet transmitted on the link, and is also the maximum possible value that the PICB count will increase by at one time. This number may be static to indicate a static buffer size, or may change after the data format has been programmed if the controller is able to vary its FIFO size based on the stream format.

3.3.41 Offset 92: {IOB}SDnFMT – Input/Output/Bidirectional Stream Descriptor *n* Format

Length: 2 bytes

Table 40. Stream Descriptor *n* Format

Bit	Type	Reset	Description
15	RO	0	*Reserved*
14	RW	0	**Sample Base Rate (BASE):** 0 = 48 kHz 1 = 44.1 kHz
13:11	RW	000	**Sample Base Rate Multiple (MULT):** 000 = 48 kHz/44.1 kHz or less 001 = x2 (96 kHz, 88.2 kHz, 32 kHz) 010 = x3 (144 kHz) 011 = x4 (192 kHz, 176.4 kHz) 100-111 = *Reserved*
10:8	RW	000	**Sample Base Rate Divisor (DIV):** 000 = Divide by 1 (48 kHz, 44.1 kHz) 001 = Divide by 2 (24 kHz, 22.05 kHz) 010 = Divide by 3 (16 kHz, 32 kHz) 011 = Divide by 4 (11.025 kHz) 100 = Divide by 5 (9.6 kHz) 101 = Divide by 6 (8 kHz) 110 = Divide by 7 111 = Divide by 8 (6 kHz)
7	RsvP	0's	*Reserved*

Bit	Type	Reset	Description
6:4	RW	00	**Bits per Sample (BITS):** 000 = 8 bits. The data will be packed in memory in 8-bit containers on 16-bit boundaries. 001 = 16 bits. The data will be packed in memory in 16-bit containers on 16-bit boundaries. 010 = 20 bits. The data will be packed in memory in 32-bit containers on 32-bit boundaries. 011 = 24 bits. The data will be packed in memory in 32-bit containers on 32-bit boundaries. 100 = 32 bits. The data will be packed in memory in 32-bit containers on 32-bit boundaries. 101-111 = *Reserved*
3:0	RW	0000	**Number of Channels (CHAN):** Number of channels in each frame of the stream: 0000 = 1 0001 = 2 … 1111 = 16

3.3.42 Offset 98h: {IOB}SDnBDPL – Input/Output/Bidirectional Stream Descriptor *n* BDL Pointer Lower Base Address

Length: 4 bytes

Table 41. Stream Descriptor *n* Lower Base Address

Bit	Type	Reset	Description
31:7	RW	0's	**Buffer Descriptor List Lower Base Address (BDLLBASE):** Lower address of the Buffer Descriptor List. This register field must not be written when the DMA engine is running or the DMA transfer may be corrupted. This value should not be modified except when the RUN bit is 0.
6:0	RO	0's	Hardwired to 0 to force 128-byte alignment of the BDL.

3.3.43 Offset 9Ch: {IOB}SDnBDPU – Input/Output/Bidirectional Stream Descriptor *n* BDL Pointer Upper Base Address

Length: 4 bytes

Table 42. Stream Descriptor *n* Upper Base Address

Bit	Type	Reset	Description
31:0	RW	0's	**Buffer Descriptor List Upper Base Address (BDLUBASE):** Upper 32-bit address of the Buffer Descriptor List. This register field must not be written when the DMA engine is running or the DMA transfer may be corrupted. This value should not be modified except when the RUN bit is 0.

3.3.44 Offset 2030h: WALCLKA – Wall Clock Counter Alias

Length: 4 bytes

Table 43. Wall Clock Counter

Bit	Type	Reset	Description
31:0	RO	0000_0000h	**Wall Clock Counter Alias (Counter):** An alias of the Wall Clock Counter register at offset 30h. This is an alias of the counter register and behaves exactly the same as if the Wall Clock Counter register were being read directly.

The Alias registers are used in some programming models to allow the position registers to be mapped directly to user mode. Since the Wall Clock Alias and the following Link Position Alias registers are at an offset 2000h above the corresponding register in the normal controller register range, the positions on the logical page starting at 2000h can be mapped to use mode without exposing all of the DMA, control, status, and interrupt registers to be visible in user mode.

3.3.45 Offset 2084, 20A4, …: {IOB}SDnLICBA – Input/Output/Bidirectional Stream Descriptor *n* Link Position in Buffer Alias

Length: 4 bytes

Table 44. Link Position in Buffer *n* Alias

Bit	Type	Reset	Description
31:0	RO	0's	**Link Position in Buffer *n* Alias (LPIBA):** An alias of the Link Position In Buffer register for each Stream Descriptor. This is an alias of the counter register and behaves exactly the same as if the Link Position register were being read directly. Note that all of the Link Position In Buffer registers for all of the supported input, output, and bidirectional stream engines are also aliased at an offset 2000h higher; e.g., Stream 0 Link Position In Buffer is aliased at 2084h, Stream 1 LPIB is aliased at 20A4h, etc.

3.4 Immediate Command Input and Output Registers

The Immediate Command Output and Immediate Command Input registers are optional registers which provide a Programmed I/O (PIO) interface for sending verbs and receiving responses from codecs. These registers can be implemented in platforms not suited for DMA command operations. If implemented, these registers must not be used at the same time as the CORB and RIRB command/response mechanisms, as the operations will conflict.

3.4.1 Offset 60h: Immediate Command Output Interface

Length: 4 bytes

Table 45. Immediate Command Output Interface

Bit	Type	Reset	Description
31:0	R/W	0's	**Immediate Command Write (ICW):** The value written into this register is sent out over the link during the next available frame. Software must ensure that the ICB bit in the Immediate Command Status register is clear before writing a value into this register or undefined behavior will result. Reads from this register will always return 0's.

3.4.2 Offset 64h: Immediate Response Input Interface

Length: 4 bytes

Table 46. Immediate Command Input Interface

Bit	Type	Reset	Description
31:0	R/W	0's	**Immediate Response Read (IRR):** The value in this register latches the last response to come in over the link. If multiple codecs responded in the same frame, there is no way to determine which response will be saved here, but the address of the codec is indicated in the ICRADD field of the Immediate Command Status register.

3.4.3 Offset 68h: Immediate Command Status

Length: 2 bytes

Table 47. Immediate Command Status

Bit	Type	Reset	Description
15:8	RsvdZ	0's	*Reserved*
7:4	RO	0's	**Immediate Response Result Address (IRRADD):** The address of the codec which sent the response currently latched into the Immediate Response Input register.
3	RO	0's	**Immediate Response Result Unsolicited (IRRUNSOL):** Indicates whether the response latched in the Immediate Response Input register is a solicited or unsolicited response.
2	RsvdZ	0's	*Reserved*
1	RW1C	0's	**Immediate Result Valid (IRV):** This bit is set to a 1 by hardware when a new response is latched into the IRR register. Software must clear this bit before issuing a new command by writing a one to it so that the software may determine when a new response has arrived.
0	RO	0's	**Immediate Command Busy (ICB):** This bit is a 0 when the controller can accept an immediate command. Software must wait for this bit to be 0 before writing a value in the ICW register. This bit will be clear (indicating "ready") when the following conditions are met: (1) the link is running, (2) the CORB is not active (CORBRP = CORBWP or CORBEN is not set), and (3) there is not an immediate command already in the queue waiting to be sent. Because software controls each of these conditions, this bit will not transition to a 1 after being read a 0 without explicit software operations to cause one of the above conditions to change.

3.5 Interrupt Structure

The Controller interrupt generation has three layers. At the bottom layer, the individual events such as buffer completion or error events cause the lowest level status indicators to be set. If the associated interrupt enable bits for these status bits are set, the interrupt will propagate up the tree to the stream or controller level, where SIE (Stream Interrupt Enable) or CIE (Controller Interrupt Enable) bit will gate interrupt generation for entire blocks of the controller. The stream and controller interrupts are collected at the Global level, where the Global Interrupt Enable bit gates interrupt generation for the entire controller.

The PCI Interrupt Disable bit in the configuration space is a further gate to interrupt generation which is generally controlled by the operating system rather than the High Definition Audio controller software driver. The interrupt generated by the controller can be either a legacy (level) PCI interrupt, or a Message Signaled Interrupt. Which type of interrupt is generated is dependent on the system implementation, and the details of implementing a MSI interrupt are not within the scope of this specification.

Figure 4 is a representation of the interrupt tree to show the logical relationship of various signals. It should not be taken as a literal implementation.

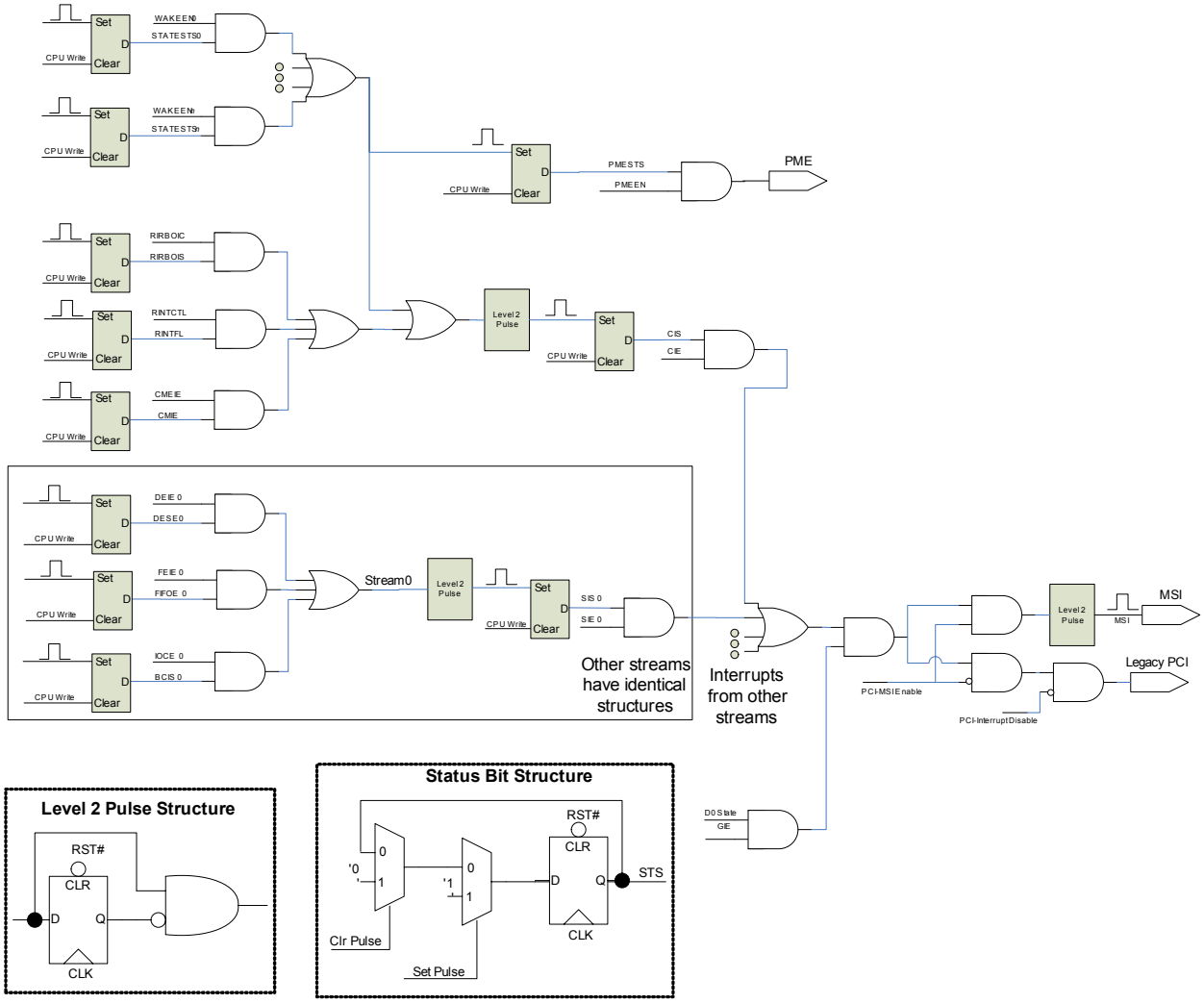

Figure 4: Controller Interrupt Structure

3.6 Data Structures

3.6.1 DMA Position in Current Buffer

The "DMA Position in Buffer" structure is written to a memory buffer each time a stream DMA position changes. Software can read this structure to determine the current stream DMA position. The structure is organized by Dwords (32 bits).

The structure must be allocated on a 128-byte boundary (the bottom 7 bits of the base address must be 0).

Table 48. DMA Position in Current Buffer

Offset (Dwords)	Description
0h	Stream Descriptor 0 Position
1h	*Reserved*
2h	Stream Descriptor 1 Position
3h	*Reserved*
…	
2*n*h	Stream Descriptor *n* Position
2*n* +1h	*Reserved*

3.6.2 Buffer Descriptor List

The Buffer Descriptor List (BDL) is a memory structure that describes the buffers in memory. The BDL is comprised of a series of Buffer Descriptor List Entries. There must be at least two entries in the list, with a maximum of 256 entries. Also, the start of the structure must be aligned on a 128 byte boundary. The BDL should not be modified unless the RUN bit is 0.

Table 49. Buffer Descriptor

Offset (Bytes)	Length (Bytes)	Field	Description
0x00	16	BDLE0	Buffer Descriptor List Entry 0
0x10	16	BDLE1	Buffer Descriptor List Entry 1
…	…	…	
0xn0	16	BDLE*n*	Buffer Descriptor List Entry *n*

3.6.3 Buffer Descriptor List Entry

Each Buffer Descriptor List Entry (BDLE) contains a description of a buffer which is a piece of the whole cyclic stream buffer. The BDLE contains a pointer to the physical memory containing the buffer, the length of the buffer, and a flag which indicates whether or not an interrupt should be generated when the buffer is complete.

The buffers described by the BDLE must start on a 128-byte boundary, and the length must be an integer number of Words.

Table 50. Buffer Descriptor

Offset (Bytes:bits)	Length (Bytes)	Field	Description
0x00	64	ADDRESS	The 64 bit address of the buffer described. The Buffer starting address must be 128-byte aligned.
0x08	32	LENGTH	The length of the buffer described in bytes. The buffer length must be at least one Word.
0x0C:0	1	IOC	Interrupt on Completion. If 1, the controller will generate an interrupt when the last byte of the buffer has been fetched by the DMA engine (if enabled by the stream's Interrupt On Completion Enable bit).
0x0C:1	31	*Reserved*	Reserved; must be 0.

3.6.4 Command Output Ring Buffer

The Command Output Ring Buffer (CORB) is a memory structure which contains the command Verbs to be sent to the codecs. The length of this structure is determined by the CORBSIZE register (see Section 3.3.24). This buffer must start on a 128-byte boundary.

Table 51. Command Output Ring Buffer

Offset (Bytes)	Length (Bytes)	Field	Description
0x00	4	VERB0	Verb 0
0x04	4	VERB1	Verb 1
…	…	…	
0xyy	4	VERB*n*	Verb *n*

3.6.5 Response Input Ring Buffer

The Response Input Ring Buffer (RIRB) is a memory structure which contains the responses, both solicited and unsolicited, from the codec. The length of this structure is determined by the RIRBSIZE register (see Section 3.3.31). This buffer must start on a 128-byte boundary.

Table 52. Response Input Ring Buffer

Offset (Bytes)	Length (Bytes)	Field	Description
0x00	4	RESP0	Response 0 is the response data received from the codec.
0x04	4	RESP0_ex	Response 0 Extended contains information added to the response by the controller. Bits 3:0 are the SDATA_INx line on which the response was received; this will correspond to the codec address. Bit 4 is a bit indicating whether the response is a solicited response (0) or an unsolicited response (1).
0x08	4	RESP1	Response 1

Offset (Bytes)	Length (Bytes)	Field	Description
0x0C	4	RESP1_ex	Response 1 Extended
…	…	…	
0xyy	4	RESP*n*	Response *n*
0xyy+4	4	RESP*n* _ex	Response *n* Extended

3.7 Codec Verb and Response Structures

The codec verb structure is entirely opaque to the controller and link, and all fields, including the address, are only interpreted by the codec.

The controller generated (outbound) Verb format is shown in Figure 5.

Bits 31 : 28	27 : 20	19:0
Codec Address	Verb	Payload

Figure 5. Verb Format

Solicited Responses from codecs are returned by the codec in response to a command Verb. All 32 bits of the Solicited Responses are opaque to the controller and link.

The Solicited Response format is shown in Figure 6.

31:0
Response

Figure 6. Solicited Response Format

Unsolicited responses are sent by the codec independently of any software request. The 6-bit "Tag" field is opaque to the controller and used by software to distinguish what codec subunit generated the Unsolicited Response.

The Unsolicited Response format is shown in Figure 7.

31:26	27:0
Tag	Response

Figure 7. Unsolicited Response Format

3.7.1 Stream Format Structure

Format is a standard structure used in the Stream Descriptors and sent to the codec. This structure does not directly appear any place in memory.

If the TYPE is set to Non-PCM, the controller just pushes data over the link and is not concerned with formatting. The base rate, data type, and number of Words (MULT) to send each valid frame are specified to control the rate at which the non-PCM data is sent.

Table 53. PCM Format Structure

Bit	Description
15	**Stream Type (TYPE):** If TYPE is non-zero, the other bits in the format structure have other meanings. 0: PCM 1: Non-PCM
14	**Sample Base Rate (BASE):** 0 = 48 kHz 1 = 44.1 kHz
13:11	**Sample Base Rate Multiple (MULT):** 000 = 48 kHz/44.1 kHz or less 001 = x2 (96 kHz, 88.2 kHz, 32 kHz) 010 = x3 (144 kHz) 011 = x4 (192 kHz, 176.4 kHz) 100-111 = *Reserved*
10:8	**Sample Base Rate Divisor (DIV):** 000 = Divide by 1 (48 kHz, 44.1 kHz) 001 = Divide by 2 (24 kHz, 22.05 kHz) 010 = Divide by 3 (16 kHz, 32 kHz) 011 = Divide by 4 (11.025 kHz) 100 = Divide by 5 (9.6 kHz) 101 = Divide by 6 (8 kHz) 110 = Divide by 7 111 = Divide by 8 (6 kHz)
7	*Reserved*
6:4	**Bits per Sample (BITS):** Number of bits in each sample: 000 = 8 bits. The data will be packed in memory in 8-bit containers on 16-bit boundaries. 001 = 16 bits. The data will be packed in memory in 16-bit containers on 16-bit boundaries. 010 = 20 bits. The data will be packed in memory in 32-bit containers on 32-bit boundaries. 011 = 24 bits. The data will be packed in memory in 32-bit containers on 32-bit boundaries. 100 = 32 bits. The data will be packed in memory in 32-bit containers on 32-bit boundaries. 101-111 = *Reserved*

Bit	Description
3:0	**Number of Channels (CHAN):** Number of channels in each frame of the stream: 0000 = 1 0001 = 2 … 1111 = 16

4 Programming Model

4.1 Theory of Operation

While the register interface is the concise description of the software interface to the High Definition Audio controller, the implementation and interpretation of these various bits is not always clear from the definition of the bit(s). This Programming Model chapter is a supplement to the register definitions and provides narrative and interpretation guidance for the behaviors of the bits.

The software operation of the High Definition Audio interface is divided into three categories: Codec Command and Control, Streaming Operation, and Link Initialization and Control. These three categories are described in detail in the following sections.

4.2 Controller Initialization

When the High Definition Audio controller comes out of power-up reset after power-on, all controller registers will be in their power-on default state, and the link will be inactive.

4.2.1 Configuring a PCI or PCI Express Interface

The first step in starting the controller is properly programming the PCI, PCI Express, or other system bus interface. Because this operation is specific to the controller implementation, the documentation for the specific controller should be followed. At the conclusion of this programming, the controller should be ready to transfer data on the system bus. For example, when using PCI, the Interrupt Line, Base Address, and other PCI Configuration space registers should be properly programmed.

4.2.2 Starting the High Definition Audio Controller

When the controller is first brought up, the CRST bit (Offset 08h, bit 0) will be 0 meaning that the controller is in reset. When the controller is in reset, the only bit which will accept writes is the CRST bit to take the controller out of CRST; all other registers will read their default values and writes will have no effect.

When a 1 is written to the CRST bit, the controller will go through the sequence of steps necessary to take itself out of reset. The link will be started, and state machines will initialize themselves. While the hardware is taking these steps, the CRST bit, if read, will still appear to be 0. When the initialization has been completed, a read of the CRST bit will return a 1 indicating that the controller is now ready to function. Therefore, after taking the controller out of reset, the software should wait until CRST is read as 1 before continuing.

Several of the High Definition Audio controller registers maintain their values across resets and power transitions. These include the WAKEEN bits, the STATESTS bits, and any other registers with type “RSM” (Resume). These bits should be examined if necessary and then reset (STATESTS) or programmed appropriately (WAKEEN) for proper operation.

4.3 Codec Discovery

When the link is enabled by the assertion of CRST, the codecs will detect the de-assertion of the **RESET#** signal and request a status change and enumeration by the controller. As the controller hardware detects these requests, it will provide the codecs with their unique addresses and set the controller STATESTS bits to indicate that a Status Change event was detected on the appropriate SDATA_INx signals. Software can use these bits to determine the addresses of the codecs attached to the link. A 1 in a given bit position indicates that a codec at that associated address is present. For instance, a value of 05h means that there are codecs with addresses 0 and 2 attached to the link.

The software must wait 250 μs after reading CRST as a 1 before assuming that codecs have all made status change requests and have been registered by the controller. This gives codecs sufficient time to perform self-initialization.

If software wishes to get an interrupt when new codecs are attached, such as during a mobile docking event, the software can set the CIE bit in the INTCTL register to a 1 to enable Controller interrupts which include the Status Change event. When the interrupt is received, the STATESTS bits can be examined to determine if a codec not previously identified has requested a status change.

4.4 Codec Command and Control

Once the attached codecs have been enumerated, commands can be sent to the codecs to determine their capabilities.

Codec Command and Control describes the mechanisms by which control information is sent to and received from the codecs. Command and Control data is low bandwidth, asynchronous data that is transmitted one command at a time on the Link. Timing is not ensured in any way, either inbound to the controller or outbound from the controller.

Codec Command and Control is handled by the controller via two key mechanisms, the Command Outbound Ring Buffer (CORB) and the Response Input Ring Buffer (RIRB).

Software is responsible for configuring the controller's CORB and RIRB via the CORB Control and RIRB Control registers.

4.4.1 Command Outbound Ring Buffer – CORB

The Controller utilizes the CORB mechanism to pass commands to the codecs. The CORB is a circular buffer located in system memory that is used to pass commands (verbs) from software to codecs connected to the High Definition Audio link. The controller uses DMA to fetch the outbound commands from the CORB and places them in the Command/Control bits at the start of each Link frame.

The size of the CORB is programmable to two entries (8 bytes), 16 entries (64 bytes), or 256 entries (1 KB) by using the CORBSIZE Controller register. Software is responsible for choosing a CORB size based on the CORBSZCAP field and the capabilities of the system. In general, the software should choose the 256 entries option unless the system capabilities dictate a smaller memory footprint.

Two pointers are maintained in the hardware, Write Pointer (WP) and Read Pointer (RP). WP is used by the software to indicate to the hardware the last valid command in the CORB, while the hardware uses RP to indicate to the software the last command that has been fetched. WP and RP both measure the offset into the buffer in terms of commands. Since commands are 4 bytes long, the byte offset into the CORB buffer indicated by RP or WP is WP*4 or RP*4.

To add commands to the CORB, the software places commands into the CORB at the end of the list of commands already in the list, which is at byte offset (WP + 1) * (4 bytes). When software has finished writing a new group of commands, it updates the WP to be equal to the offset of the last valid command in the buffer. When the CORB is first initialized, WP = 0, so the first command to be sent will be placed at offset (0 + 1) * 4 = 4 bytes, and WP would be updated to be 1.

When the CORB RUN bit is set, the DMA engine in the Controller will continually compare the RP to the WP to determine if new commands are present for consumption. When the Read Pointer is not equal to the Write Pointer, the DMA engine runs until the pointers match, and the fetched commands are transmitted on the link. The DMA engine reads the commands from the CORB and sends them to the Codecs over the link.

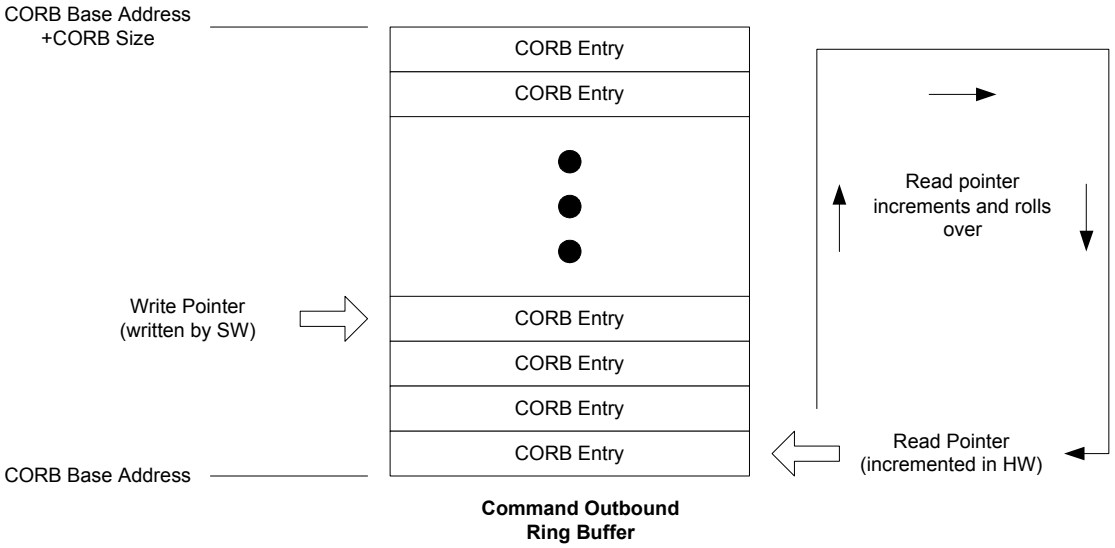

Figure 8. Command Ring Buffer (CORB)

4.4.1.1 CORB Buffer Allocation

The CORB buffer in memory must be allocated to start on a 128-byte boundary and in memory configured to match the access type being used. For instance, a PCI Express controller using non-snooped accesses should allocate and access the CORB buffer in a way that maintains coherency between the processor cache and the physical memory.

The location of the CORB is assigned by software and written to the Controller's CORB Address Upper Base and Lower Base register. The lowest 7 bits of the Lower Base Address are always 0 to enforce the 128-byte alignment requirement.

4.4.1.2 CORB Entry Format

The verbs passed from the Controller to the Codecs are 32 bits long. Each entry in the CORB is also 32 bits long and matches the verb format. The verb format is opaque to the controller hardware and only has meaning to software and the codecs.

4.4.1.3 Initializing the CORB

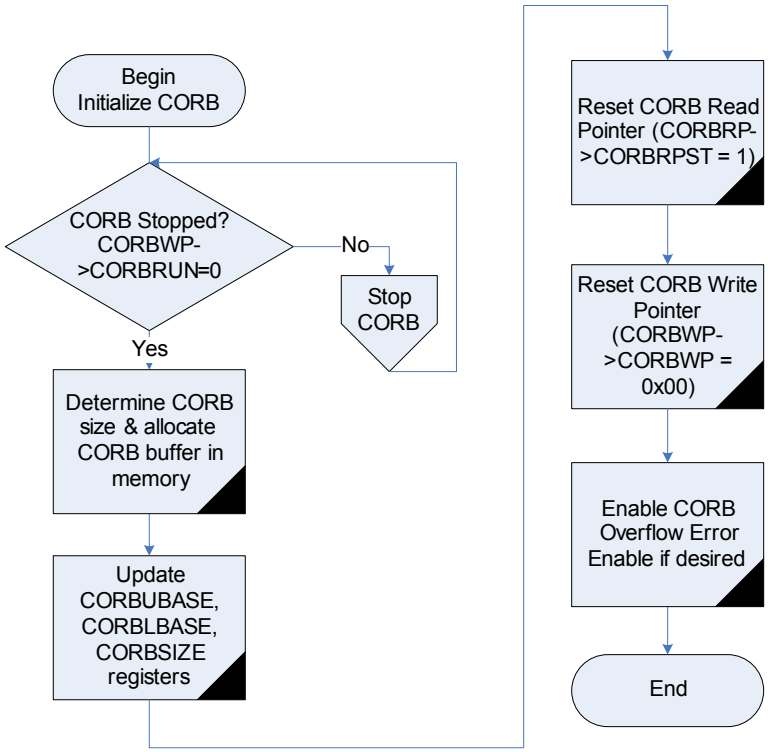

Figure 9. CORB Initialization

To initialize the CORB, first the software must make sure that the CORB is stopped by making sure that the CORBRUN bit in the CORBCTL register is 0. The correct register size is determined using the CORBSIZE register, and the CORB memory is allocated from the appropriate heap and memory type.

The CORBBASE registers are programmed to the base of the allocated memory, and the CORBRPRST bit is used to reset the Read Pointer to 0. Software must write 0h to the Write Pointer to clear the Write Pointer. If desired, CORB error reporting may be enabled by setting the CMEIE bit. Lastly, the CORBRUN bit is set to 1 to enable CORB operation.

4.4.1.4 Transmitting Commands via the CORB

Transmission of commands via the CORB begins with checking to make sure there is sufficient space in the CORB. The difference between the CORBWP and CORBRP can be examined to determine the space available in the CORB. If the block of commands is larger than can fit in the CORB, it may be necessary to break the block of commands into multiple smaller blocks to send.

The commands are written into the CORB starting at the location indicated by the index CORB WP + 1, which is the first free space for a command. Note that in the case of the first block of commands, this means that the first commands will be placed at a offset of 4 bytes into the CORB buffer, as CORBWP will be 0, so CORBWP + 1 will indicate a 4-byte offset into the CORB.

CORBWP is then updated by software to reflect the index of the last command that has been written into the CORB. Hardware will then begin to transfer the commands over the link, updating CORBRP with each command fetched from memory. All commands have been sent when CORBRP is equal to CORBWP, at which point the controller will stop sending verbs until software repeats the process and sets CORBWP to a different value.

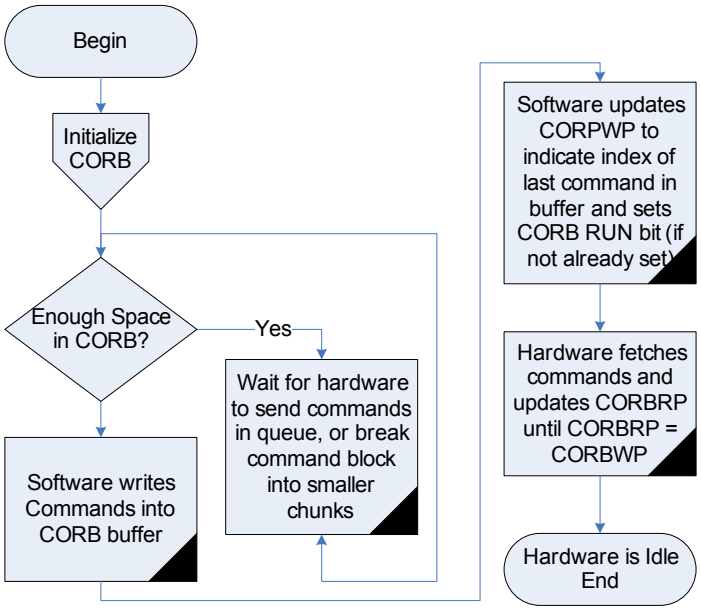

Figure 10. Transmitting Commands via the CORB

While the hardware is in the process of sending commands (CORBRP ≠ CORPWP and CORB RUN is set), software may add new commands into the CORB buffer after the index indicated by CORBWP, and then update CORBWP. Hardware must continue to send the newly added commands. Software must ensure that the newly added commands do not overflow the buffer; i.e., that no command added will overwrite commands that have not yet been sent, as indicated by the current value of the CORBRP.

4.4.1.5 Other CORB Programming Notes

If large numbers of commands are being sent to a codec, it is possible that the codec may be blocked from returning Unsolicited Responses because it is required to always respond to solicited verbs on the following frame. For this reason, it is recommended that the software occasionally insert breaks in the verbs being sent if a large block is being transmitted at one time.

If it is desired to insert breaks in the CORB, insert 00000000h commands, which are NULL commands. Since the codec will not have a solicited response to the NULL command, it provides an opportunity for the codec to respond with an unsolicited response. Note that when the software is matching responses with the commands, the NULL commands will not generate responses.

4.4.2 Response Inbound Ring Buffer - RIRB

The responses from the codecs are sent to the controller via the RIRB mechanism. The RIRB is a circular buffer located in system memory that is used to store responses from codecs connected to the Link. Responses can either be solicited (in response to a command from the controller) or unsolicited (sent by the codec to signal an event).

The size of the RIRB is programmable to two entries (16 bytes), 16 entries (128 bytes), or 256 entries (2 KB). The location of the RIRB is assigned by software and written to the Controller's RIRB Address register.

The RIRB buffer in memory must be allocated on a 128-byte boundary and in memory configured to match the access type being used. For instance, a PCI-Express controller using non-snooped accesses should allocate and access the CORB buffer in a way that maintains coherency between the processor cache and the physical memory.

A Response can be sent to the controller from any one of the codecs, and the DMA engine in the Controller writes the response to the RIRB. Solicited responses are returned by an individual codec in the subsequent frame and in the same order that the prompting commands were sent to that codec. Unsolicited responses may be injected by the codec in any frame where a solicited response is not present. The controller will write this stream of responses to the RIRB buffer. Software is responsible for separating responses from the individual codecs as well as separating unsolicited responses from solicited responses.

As with the CORB, a Read Pointer and a Write Pointer are used in the RIRB. In the RIRB, though, the RP is kept only by software to remember the last response the software read from the response buffer; there is no hardware representation of the RP. The Controller keeps a WP in hardware to indicate the offset of the last response which has been written into the response buffer. The Controller blindly writes to the RIRB whenever a response (solicited or unsolicited) is returned. As with the CORB, the WP indicates the offset in the response buffer in units of responses. Since each response is 8 bytes, the byte offset into the buffer is (WP * 8 bytes).

There are two ways for the Controller to notify software that the RIRB entries may be read:

1. Interrupt: The Controller will generate an interrupt after a programmable N number of Responses are written to the RIRB or when an empty Response slot is encountered on all SDATA_IN_X inputs, whichever occurs first. Software can then look at the Write Pointer and determine the entries added to the RIRB.
2. Polling: Software may poll the hardware Write Pointer and compare it to the software maintained Read Pointer. If they are different, entries have been added to the RIRB, and software should read RIRB entries until up to the Write Pointer. Software is responsible for polling often enough to ensure that the hardware Write Pointer does not wrap around; if it does wrap, the responses will be lost.

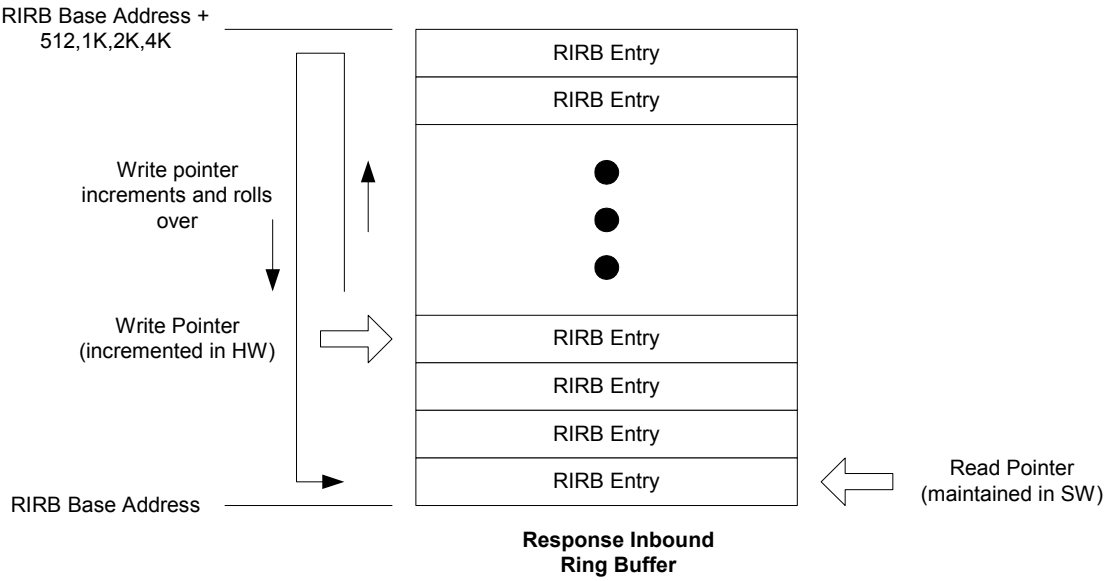

Figure 11. Response Inbound Ring Buffer

4.4.2.1 RIRB Entry Format:

The response passed from the codecs to the controller is 32 bits long. Each entry in the RIRB is 64 bits long. In addition to the 32 bits representing the actual response data from the codec, the Controller adds the following information to the RIRB entry:

- Codec # based on the SDATA_IN_x signals on which the response was received.
- Solicited versus unsolicited response indicator.

Table 54. RIRB Entry Format

Offset (Bytes)	Length (Bytes)	Field	Description
0x00	4	Response	Response is the response data received from the codec.
0x04	4	Resp_Ex	Response Extended contains information added to the response by the controller. Bits 3:0 is the codec; i.e., the SDATA_INx line on which the response was received; this will correspond to the codec address. Bit 4 is a bit indicating whether the response is a solicited response (0) or an unsolicited response (1).

The bit definitions are as follows:

Codec:

0000 = Response received on SDATA_IN_0

0001 = Response received on SDATA_IN_1

0010 = Response received on SDATA_IN_2

0011 = Response received on SDATA_IN_3

…

1110 = Response received on SDATA_IN_14

Sol/Unsol:

0 = Solicited Response

1 = Unsolicited Response

4.4.2.2 Initializing the RIRB

RIRB initialization is very similarly to CORB initialization. The memory must be allocated correctly based on the RIRBSIZE register and from the heap appropriate for the system infrastructure. The RIRBUBASE, RIRBLBASE, and interrupt generation control registers are then updated appropriately.

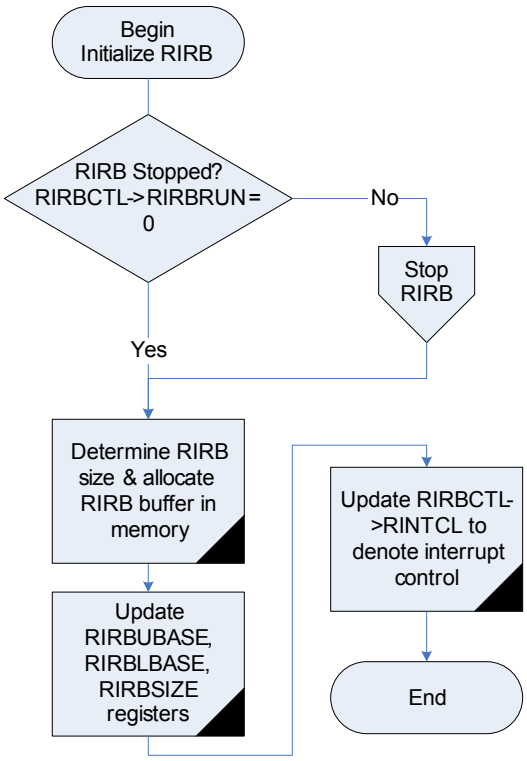

Figure 12. Initializing the RIRB

4.5 Stream Management

The High Definition Audio architecture uses the concept of streams and channels for organizing data for transmission on the Link. A stream is a virtual connection created between a system memory buffer(s) and the codec(s) rendering or capturing that data and which is driven by a single DMA channel through the link.

Software is responsible for creating and managing streams. Key parameters related to a stream such as stream parameters, number of channels, data format, bit depth, etc., are defined by software.

It is the responsibility of software to determine which converters (and associated hardware) on which codecs are associated with a given stream. For example, software may need to determine which jacks should be assigned to those converters.

4.5.1 Stream Data In Memory

Samples represent one channel of data to be played at one instant in time. In a 24 bit, three-channel, 96-kHz stream, one sample is 24 bits long. Samples are packed in *containers* which are 8 bits, 16 bits, or 32 bits wide; the smallest container size which will fit the sample size is used. In the case of a 24-bit sample, a 32-bit container would be used. Samples are padded with 0's at the LSB to left justify the sample within the container. Samples must be naturally aligned in memory. Samples in 16-bit containers must be Word aligned, and samples in 32-bit containers must be Dword aligned.

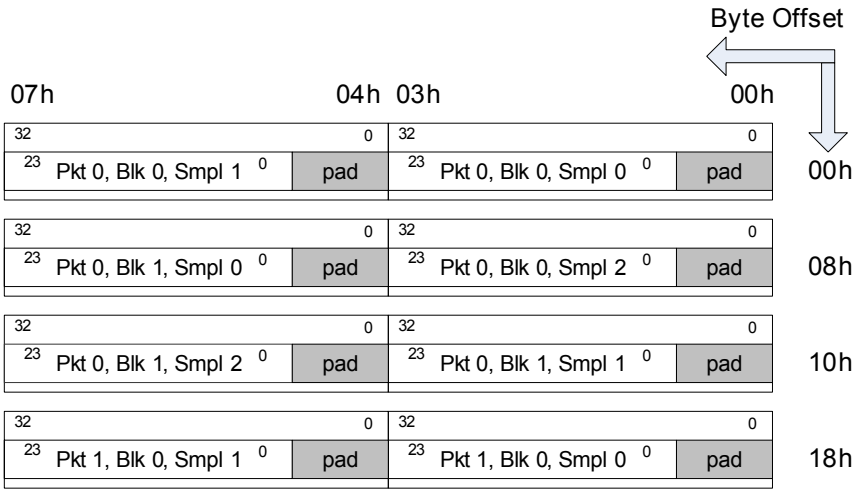

Figure 13. A 24-bit, Three-Channel, 96-kHz Stream in Memory

Blocks are sets of samples to be rendered at one point in time. A block has a size of (container size * number of channels), so the 24-bit, three-channel, 96-kHz stream would have a block size of 12 bytes in memory.

The standard output Link rate is based on a 48-kHz frame time, therefore, in the case of a stream running faster than 48 kHz, multiple blocks must be transmitted at one time. Multiple blocks transmitted at the same time are *packets*. The 24-bit, three-channel, 96-kHz stream being used as an example would have a packet size of 24 bytes.

Packets are collected in memory into *buffers*, which are commonly (but are not required to be) whole pages of memory. Each individual buffer must contain an integer number of samples, but blocks and packets may be split across multiple buffers. The buffer must start on a 128-byte boundary and must contain at least one sample of data. For highest efficiency, the following guidelines should be met in buffer allocation:

- The buffer should have a length which is a multiple of 128.
- The buffer should contain at least one full packet of information.

There may be other system-dependent guidelines that can further increase memory efficiency, but buffers that do not meet at least the above guidelines may have a negative effect on system performance as bus and memory efficiency may be significantly affected.

The set of all of the buffers for the stream taken together describe the virtual *Cyclic Buffer*, which is generally mapped by the software into a single contiguous memory region for easy access. The Buffer Descriptor List for the stream describes to the hardware the starting address and length of each buffer to be played. The Cyclic Buffer must contain an integer number of packets.

4.5.2 Configuring and Controlling Streams

4.5.3 Starting Streams

To create a stream, the software must first determine the appropriate stream parameters, such as sample rate, bit depth, and number of channels. The controller and codec resources should be checked to make sure sufficient resources are available to support the desired stream format.

A data buffer and BDL buffer are then allocated from the proper memory pool, making sure that the proper buffer alignment and caching requirements are met. The BDL is constructed to describe the stream data buffer, including setting Interrupt on Completion bits at the points interrupts are desired. Software then allocates a Stream Descriptor (Input or Output, as appropriate), and configures the Descriptor with the stream format, BDL address, Cyclic Buffer Length, interrupt policy, and other necessary register settings.

The codec is next configured by configuring the appropriate Audio Input or Output Converters with the stream ID and format information. Other widgets in the audio path are configured so that the audio data flows to (or from) the Pin Complex Widgets, with Connection Select, Amplifiers, Processing Controls, and all other controls in the audio path set as appropriate. At this point, the codec is ready to accept data if the stream is an output stream or begins sending data on the link for an input stream.

The Stream Descriptor's RUN bit is then set to 1 to start the DMA engine and begin the transfer of data to or from the link.

4.5.4 Stopping Streams

To stop a stream, the software writes a 0 to the RUN bit in the Stream Descriptor. The RUN bit will not immediately transition to a 0. Rather, the DMA engine will continue receiving or transmitting data normally for the rest of the current frame but will stop receiving or transmitting data at the beginning of the next frame. When the DMA transfer has stopped and the hardware has idled, the RUN bit will then be read as 0. The run bit should transition from a 1 to a 0 within 40 μs.

4.5.5 Resuming Streams

If a stream which was previously running has been stopped, it can be restarted by setting the RUN bit back to 1. If the stream has been recently stopped, the RUN bit must be checked to make sure that it has transition back to a 0 to indicate that the hardware is ready to restart. When the RUN bit is again set to 1, the DMA engine will then restart at the point it left off.

4.5.6 Stream Steady State Operation

Once the stream has been started, the hardware will continue fetching data from the Cyclic Buffer described by the Buffer Descriptor List. For an output stream, software is responsible for making sure that valid data is present in the buffers before it is fetched by the hardware. For an input stream, data must be removed from the buffers before being overwritten by hardware. When the hardware has reached the end of the Cyclic Buffer, it will automatically wrap back around to the beginning of the buffer and continue to process the stream data until the stream is stopped by the software by clearing the RUN bit.

Software can either use interrupts at the end of selected buffers by setting the IOC bit in the BDL entry or can poll the stream position to determine when to process the stream data.

If interrupts are being used, some care must be taken to access the stream Status and Control registers in a safe manner. A recommended policy is that the Interrupt Service Routine only use byte access to read or write the Status register to clear the status bits. The ISR should not attempt to write to the stream Control register, as there may be synchronization issues between the ISR and the non-ISR code both trying to perform Read-Modify-Write cycles on the register.

After the RUN bit has been set, the buffer described by the BDL should not be changed by the software. The hardware may pre-fetch and/or cache an arbitrary number of BDL Entries from the list, so there is no way to ensure when or if any changes to the BDL list would be visible to the hardware. Even when the RUN bit has been cleared to pause the stream after it has been running, the hardware may still have pre-fetched descriptors that will not be flushed when the stream is restarted. Therefore, the software should only modify the BDL before the RUN bit has been set for the first time after a Stream Reset.

4.5.7 Synchronization

There are three different domains in which synchronization of streams is desired. They are multiple streams on different systems, multiple streams on different controllers in the same system, and multiple streams on the same controller.

The High Definition Audio Specification does not provide any mechanisms to aid in synchronizing streams on two different systems but does provide mechanisms to synchronize streams within a system. The wall clock can be used to synchronize between two separate controllers which do not share a common clock, and the stream start synchronization can be used to synchronize exactly two streams on the same controller.

4.5.7.1 Controller to Controller Synchronization

The High Definition Audio Specification requires that the controller provide a “wall” clock, implemented in the Global Synchronization and Control Register, which is a monotonically increasing 32-bit counter. Whenever the bit clock is running, this clock is also running. This time

base may be used to account for drift between two different audio subsystems by performing micro-sample rate conversion operations on the audio data to keep the drift between streams running on the independent controllers to within a specified error margin.

4.5.7.2 Stream to Stream Start Synchronization

Using the SSYNC bits in combination with the stream RUN bits, multiple input and streams can be synchronized in time.

When the hardware is initialized, all the relevant stream descriptor's RUN bits are cleared. Also, the relevant SSYNC bits are cleared as well. To synchronously start a set of streams, software will set the relevant SSYNC bits to 1 to indicate the set of streams to be synchronously started and then set the stream's RUN bit to cause the DMA engine to fetch data. While the SSYNC bits are set, though, data will not be sent on to the link, essentially leaving the stream in a "pause" state.

Software then must wait until each output stream's FIFORDY bit is set, indicating that the DMA engines have fetched enough data to start the stream in the case of an output stream, or fetched enough buffer descriptors to have a place to put the incoming data in the case of an input stream. This ensures that all output streams will have sufficient data ready, or a place to put it, when the streams are started. Note that for input streams, FIFORDY is set to 1 as a function of whether one or more Input-Descriptors are available in the Input Stream buffer, independent of the descriptor's length value. If the first descriptor-length value is very small, it increases the likelihood that overrun condition will occur.

When all relevant FIFORDY bits are set, software clears the relevant SSYNC bits using a single write to the Stream Synchronization register causing the streams to begin flowing. All output streams will transmit their first sample on the link frame following the de-assertion of their SSYNC bit, and input streams will capture data from the link frame following the de-assertion of their SSYNC bit.

4.5.7.3 Stream to Stream Stop Synchronization

The sequence starts while relevant streams are actively receiving (input) or transmitting (output). Their respective RUN bits are set to 1 while SSYNC bits are cleared to 0.

To begin the synchronized stop, software writes a 1 to the relevant SSYNC bits. As a result of the SSYNC bits being set, the controller will stop receiving (input) or transmitting (output) in the beginning of the next frame for the relevant streams. Software may then clear each individual streams' RUN bits to 0 to stop the stream's DMA engine.

Once software detects that all relevant RUN bits are cleared to 0, it can clear the SSYNC bits to a 0 to return the controller to the initial ready state. Software can then restart the streams using the same sequence described in Section 4.5.7.2, or it may start them individually without using the SSYNC bits.

4.5.8 Power Management

4.5.8.1 Power State Transitions

When the controller or codecs are transitioned to a D0 state from a D3 state, it is possible that the hardware may have had power removed, and, therefore, software must not assume that registers retain values previously programmed, with the exception of controller registers marked "RSM." Software should therefore restore all codec registers on a transition to D0 from D3.

4.5.8.2 Power Optimization

While there is no requirement that software perform power optimizations, in many environments power savings are desired. Software may optimize at three levels. The easiest power management level is using the CRST bit in the controller to power up or down the entire controller and link at once. This method is obviously not always suitable, as any activity on the link to any codec will prevent software from entering this state. This state also takes the longest to resume from, as the entire subsystem including the controller and codec must be reinitialized before a stream can be started.

Software may also control power at the Function Group level by using the Power State Control at the Function Group level. This method is generally sufficient for most systems, especially where there are multiple function groups on the link, such as audio and modem. In this case, function groups not in use may be shut down without affecting the operation of other function groups.

To achieve optimal power conservation, software may use widget level power controls to shut down widgets not in use. In the example case of an audio function group, software may be able to shut down amplifiers and other power consuming components in the codecs without affecting active streams using other paths in the codec. The actual level of power savings may vary considerably, as different codec hardware implementations may make different power usage versus complexity tradeoffs.

4.5.9 Codec Wake

4.5.9.1 Codec Wake From System S0, Controller D0

When the system is in S0 and the controller is active, a codec will use an unsolicited response to indicate to software that it requires attention. For instance, a Modem Function Group may use a vendor defined unsolicited response on "Ring Indicate" to request attention from software. Software can then use the source of the unsolicited response and the Unsolicited Response Tag to know which Function Group requires service. The Audio Function Group definition does not currently define an unsolicited response to indicate a "Wake" situation; such a mechanism would be vendor defined.

4.5.9.2 Codec Wake From System S0, Controller D3

When the controller is in D3 (CRST is set to a 1), the link will not be running, and codecs will use a power state change request on the link (see Section 5.6) to indicate to the controller that they require service. Software can control which codecs may wake the system by setting the WAKEEN bits appropriately. On a wake event, software reads the STATESTS register (before taking the controller out of reset) to determine which codec(s) have requested a power state change. Software

must then further query the function groups in the codec to determine which function group requested the wake service.

4.5.9.3 Codec Wake From System S3

When the system is in S3 at the time a codec issues a power state change request and the associated bit in the WAKEEN registers is set, the hardware will route the request to the system PME logic and the ACPI subsystem (or other power management mechanism as implemented in the system). This will cause the system to transition to S0. Software can control which codecs may wake the system by setting the WAKEEN bits appropriately. When software regains control, it can examine the STATESTS bits to determine which codec(s) have requested power state transitions and handle as necessary. Software must then further query the function groups in the codec to determine which function group requested the wake service.

4.5.9.4 Checking Wake Status on Resume

In the link bring up sequence, there is a time during the codec initialization when codecs may neither generate a power state change request on the link nor generate unsolicited responses. If it is vital that a wake event not be missed, then software should check the function group on any controller or codec transition from D3 to D0 to determine whether the function group has requested a wake. In most cases, this is not necessary as with a modem where the next ring will cause the necessary notification if the first notification is lost during the transition from D3 to D0. The Function Group control to check will be vendor unique and is not defined for the Audio Function Group.

5 Link Protocol

5.1 Introduction

This chapter describes the logical operation of the High Definition Audio Link, excluding the electrical and absolute timing characteristics, which are described in Chapter 6. This chapter contains, in logical order:

- Signal definitions, connection topologies, and relative timing
- Message composition on the link
- Stream independent, multiple sampling rates
- Link reset and initialization
- Power management behavior

5.2 Link Signaling

The High Definition Audio link is the digital serial interface that connects High Definition Audio codecs to the High Definition Audio controller. The link protocol is controller synchronous, based on a fixed 24.00-MHz clock and is purely isochronous (no flow control) with a 48-kHz framing period. Separate input and output serial digital signals support multiple inbound and outbound streams, as well as fixed command and response channels.

Figure 14 shows the key concepts of link functionality.

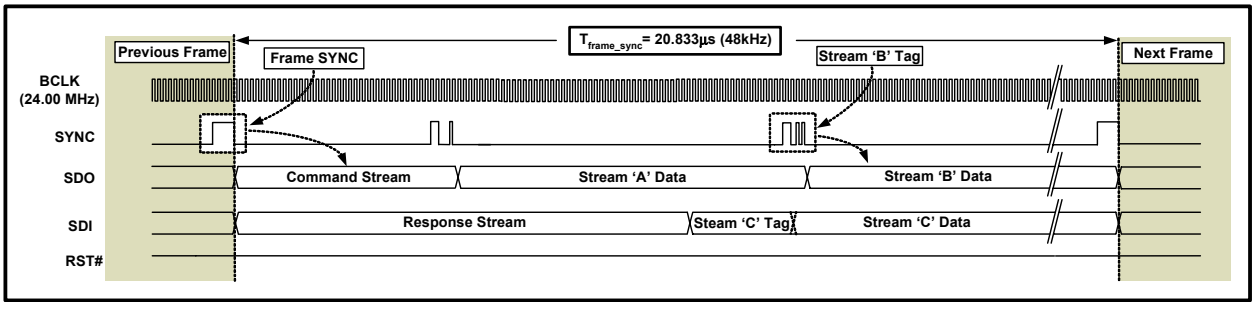

Figure 14. High Definition Audio Link Conceptual View

5.2.1 Signal Definitions

The signals required to implement a High Definition Audio link between a High Definition Audio controller and High Definition Audio codec are summarized in Table 55 and the following definitions.

Table 55. High Definition Audio Link Signal Descriptions

Signal Name	Source	Type	Description
BCLK	Controller	O	Global Link 24.00-MHz clock
SYNC	Controller	O	Global 48 kHz Frame Sync and outbound tag signal
SDO	Controller	O	Bussed Serial Data Output(s)
SDI	Codec and Controller	I/O - PD	Point-to-point Serial Data Input(s). Controller has a weak pull down
RST#	Controller	O	Global active low reset

BCLK – Bit Clock: 24.00-MHz clock sourced from the controller and connecting to all codecs on the Link.

SYNC – This signal marks input and output frame boundaries (Frame Sync) as well as identifying outbound data streams (stream tags). **SYNC** is always sourced from the controller and connects to all codecs on the link.

SDO – Serial Data Out: one or more serial data output signal(s) driven by the Controller to all codecs on the link. Data is double pumped – i.e., the controller drives data onto **SDO**, and codecs sample data present on **SDO** with respect to every edge of **BCLK**. Controllers must support at least one **SDO** and may support extra **SDO** lines for extended outbound bandwidth. Multiple **SDO**s must be implemented in powers of 2 (1, 2, or 4). In this chapter, **SDO** refers to all **SDO** signals collectively; specific **SDO** signals will always be referenced with a subscript.

SDI – Serial Data In: one or more point-to-point serial data input signals, each driven by only one codec. Data is single pumped; codecs drive **SDI** and the controller samples **SDI** with respect to the rising edge of **BCLK**. Controllers are required to support at least one **SDI** signal. In this chapter, **SDI** refers to all **SDI** signals collectively; specific **SDI** signals will always be referenced with a subscript. Controllers are required to support weak pulldowns on all **SDI** signals. These pulldowns are active whenever the controller is powered or in a wake enabled state. **SDI** pulldowns are required to prevent spurious wake event in electrically noisy environments.

RST# - Active low link reset signal. **RST#** is sourced from the controller and connects to all Codecs on the link. Assertion of **RST#** results in all link interface logic being reset to default power on state.

5.2.2 Signaling Topology

BCLK, **SYNC**, and **RST#** are shared in common by the controller and all attached codecs. These provide for basic signal timing and initialization. The Serial Data Out (**SDO**) signal(s) are multi-drop attaching to all codecs, while Serial Data In (**SDI**) signal(s) are point-to-point between a given codec and the controller. Both **SDO** and **SDI** are separately expandable, allowing link bandwidth to scale. Basic systems have one **SDO**, plus one **SDI** for each attached codec. Systems with higher performance/function codecs may have multiple **SDO**'s and/or multiple **SDI**'s.

5.2.2.1 Basic System

Figure 15 shows how a Controller and its associated codecs are connected. Note that **BCLK**, **SYNC**, and **RST#**, all driven by the controller, are connected as a single multi-drop network. This figure also shows a single **SDO** signal (driven by the controller) connected to all codecs. In addition, each codec has its own point-to-point **SDI** signal connected separately to the controller. The High Definition Audio Architecture supports up to 15 codecs thus connected, although electrical constraints and product requirements may constrain that number.

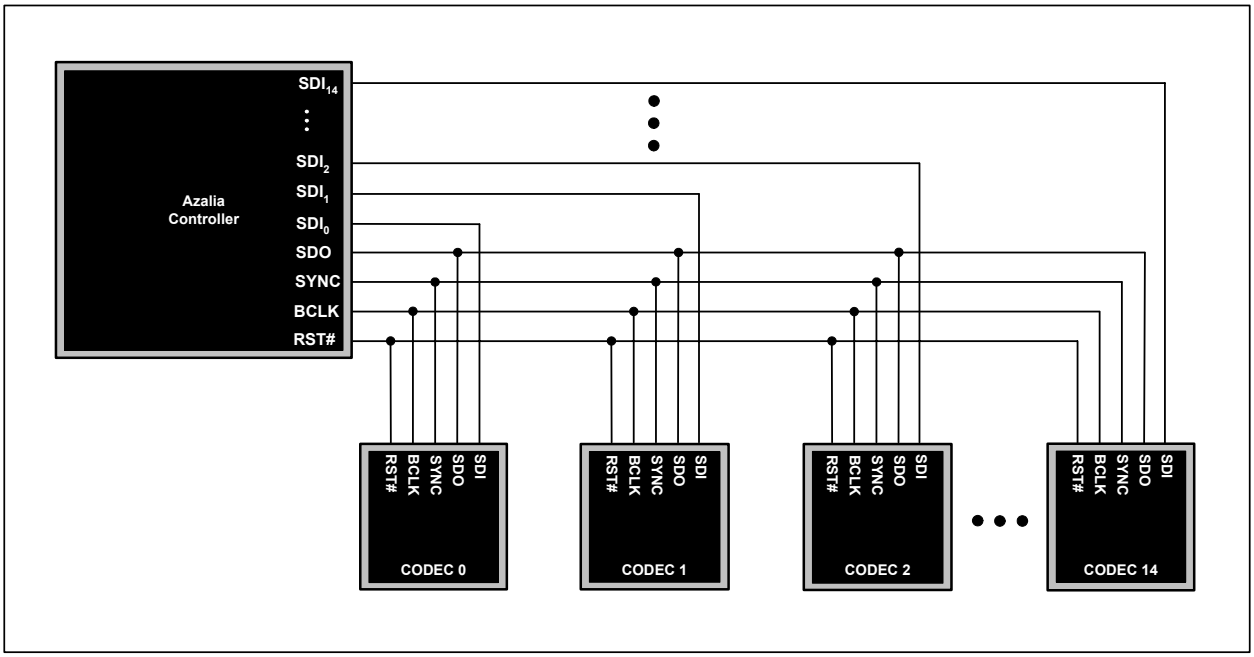

Figure 15. Basic High Definition Audio System

5.2.2.2 Bandwidth Scaling

The High Definition Audio Architecture provides for inbound and outbound bandwidth scaling by allowing individual codecs to connect to multiple **SDO** and/or **SDI** signals, as shown in Figure 16. However, **SDO**$_0$ and at least one **SDI** connection are required connections for all codecs on the link. This ensures that the command and response functionality is preserved independent of higher order **SDO** and **SDI** connection to the controller.

In this example, both "Codec 0" and "Codec N" use two **SDO** signals for extra output bandwidth. Such codecs should only be used with a controller providing two or more **SDO** signals. Failure to connect all of a codec's **SDO**s may result in reduced codec capabilities depending on the codec design. See Section 5.5.3.4 for more information on partially or un-initialized codec behavior. "Codec 1," not requiring extra bandwidth, connects only to **SDO**$_0$, which is required of all codecs.

Figure 16 also shows Codec 1" and "Codec N" each connecting to two **SDI** signals, whereas codec 0 uses only one **SDI** signal. Codecs with multiple **SDI** signals should have all of these connected to the controller. Failure to connect to all **SDI** signals to the controller may result in reduced codec capabilities, depending on the codec design. See Section 5.5.3.4 for more information on partially or un-initialized codec behavior. In all cases, codecs which support

multiple **SDO** or **SDI** connections must be capable of properly receiving and responding to command and control operation (verbs and responses), whether all **SDO**s and **SDI**s are connected, or only $\mathbf{SDO}_0$ and a single **SDI** are connected.

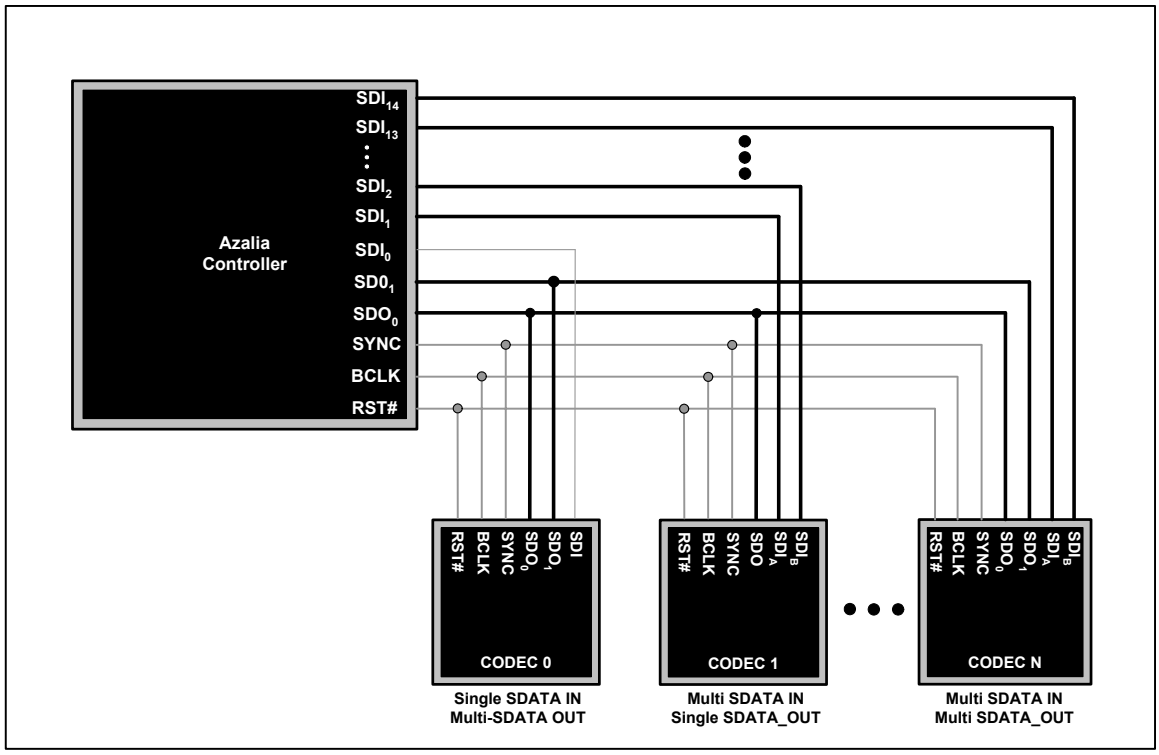

Figure 16. Serial Data Bandwidth Scaling

5.2.3 Relative Signal Timing

The High Definition Audio Link defines 500 input bit cells on **SDI** and 1000 output bit cells on **SDO** in each isochronous frame. Figure 17 shows these bit streams, numbered from 499 to 0 and from 999 to 0 respectively. It also shows the "double pumped" nature of **SDO**, and that bit 499 on **SDI** aligns with bits 999 and 998 on **SDO**, all beginning with the falling edge of Frame Sync on **SYNC**, which marks the beginning of a new frame. The exact timing details are of course dependent on various circuit delays, which are all specified in Chapter 6 of this specification. This section only identifies relative timing and the reference clock edges for these signals.

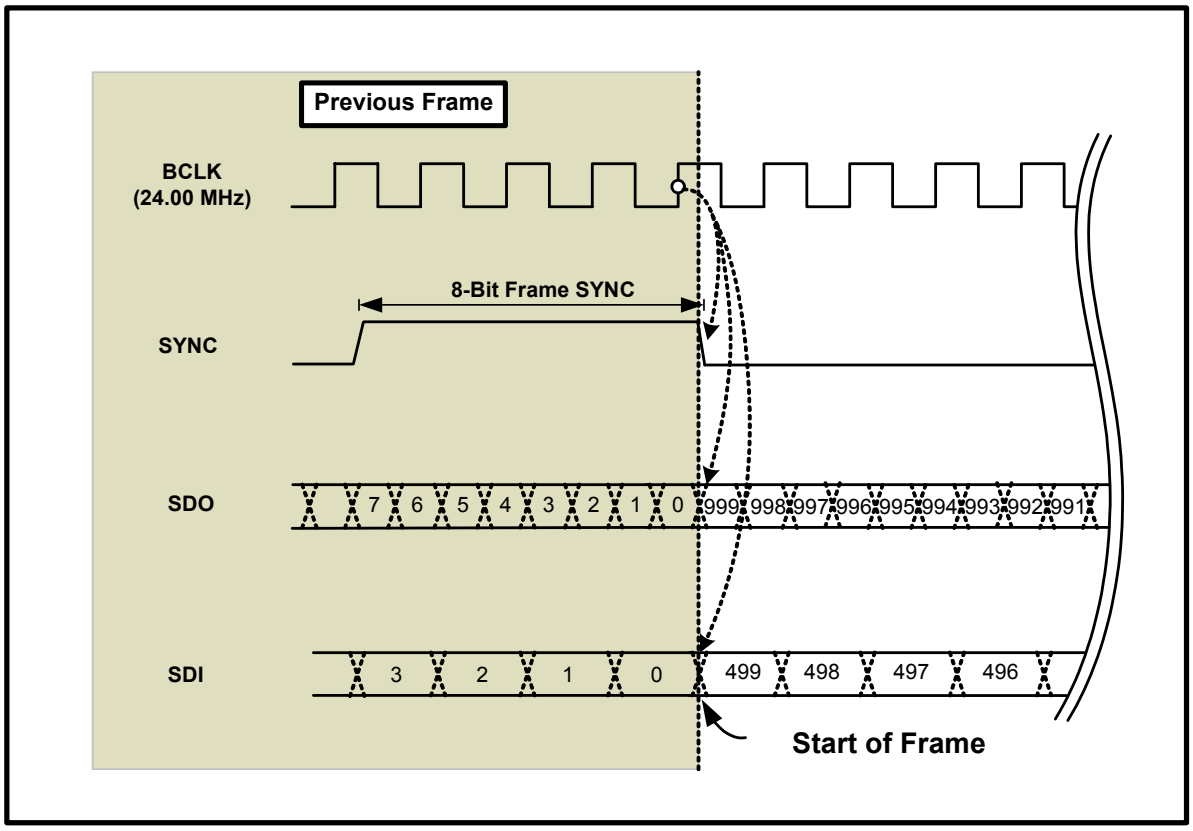

Figure 17. SDO and SDI Bit Timing

The timing of **SDI**, **SDO**, and **SYNC** are all defined with respect to certain edges of **BCLK**. Note that Figure 17 shows that output bit cell 999 and input bit cell 499, as well as the falling edge of Frame Sync, are all driven relative to a rising edge of **BCLK**.

Figure 18 shows that both **SDO** and **SYNC** may be toggled with respect to either edge of **BCLK**. In particular, bit cell *n+1* is driven by the controller on **SDO** with respect to clock edge #2, and is sampled by the codec with respect to the subsequent clock edge, #3, and so forth.

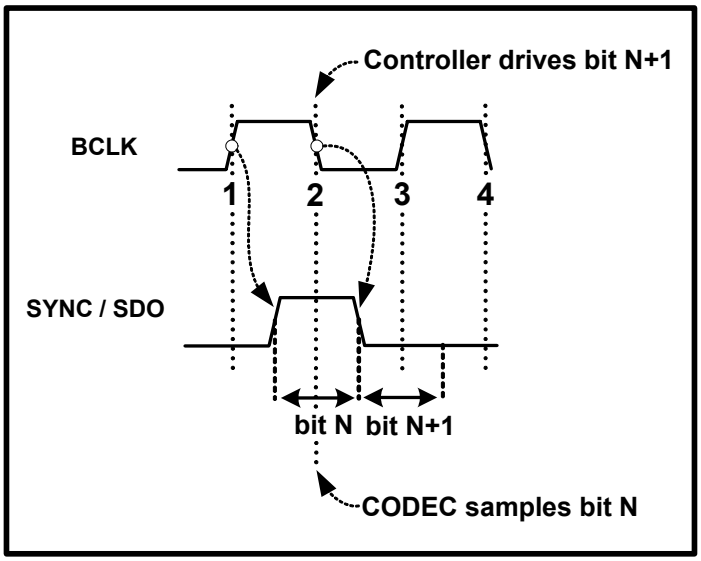

Figure 18. SYNC and SDO Timing Relative to BCLK

Figure 19 shows that **SDI** may only be toggled with respect to the rising edge of **BCLK**. In particular, bit cell *n+1* is driven by the codec on **SDI** with respect to rising clock edge #2 and is sampled by the controller with respect to the subsequent rising clock edge, #3, and so forth.

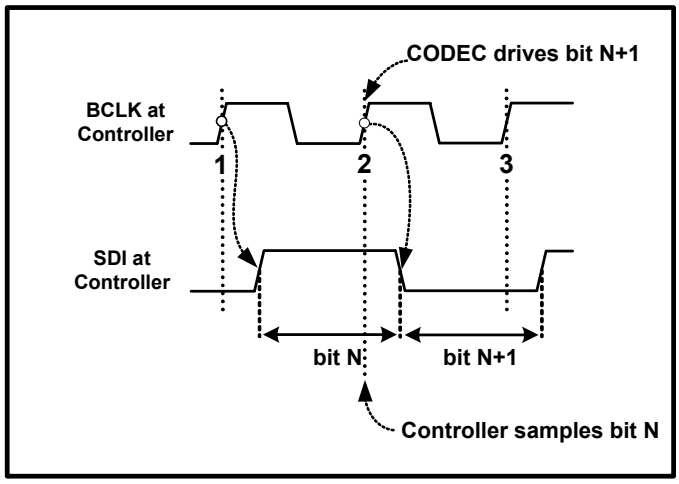

Figure 19. SDI Timing Relative to BCLK

5.3 Frame Composition

Since the link is purely an isochronous transport mechanism, all link data transmission occurs within periodic time frames.

A *frame* is defined as a 20.833-µs window of time marked by the falling edge of the Frame Sync marker, which identifies the start of each frame. The controller is responsible for generating the Frame Sync marker, which is a high-going pulse on **SYNC**, exactly four **BCLK** cycles (eight **SDO** bit times) in width, as shown in Figure 20.

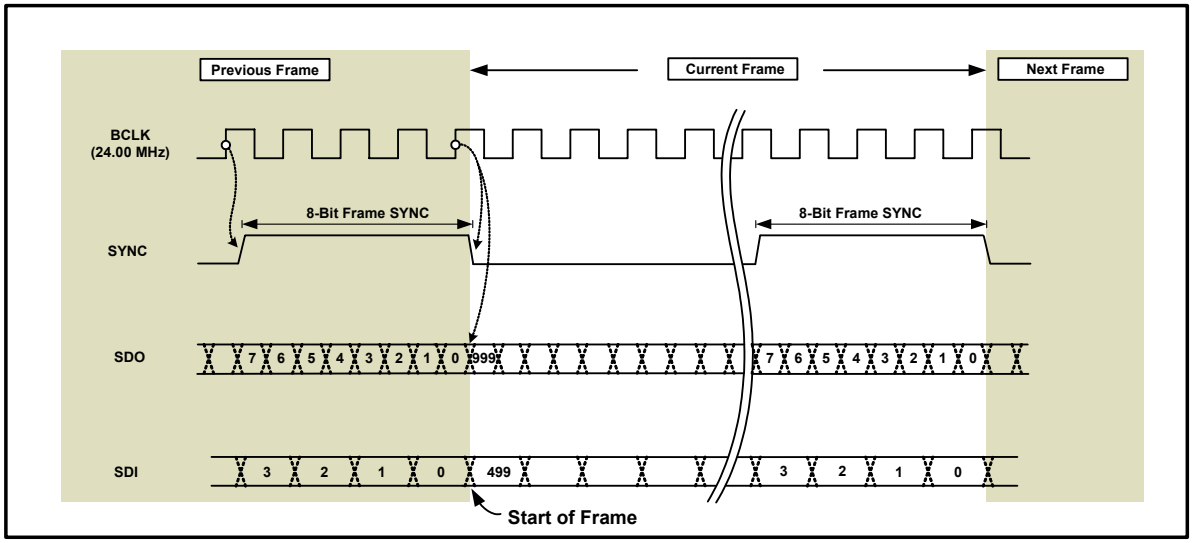

Figure 20. Frames Demarcation

5.3.1 Basic Frame Components

Inbound and outbound frames are made up of three major components, specifically:

- A single *Command/Response Field*
- Zero or more *Stream Packets*
- A *Null Field* to fill out the frame

All of these are shown in Figure 21 and described below.

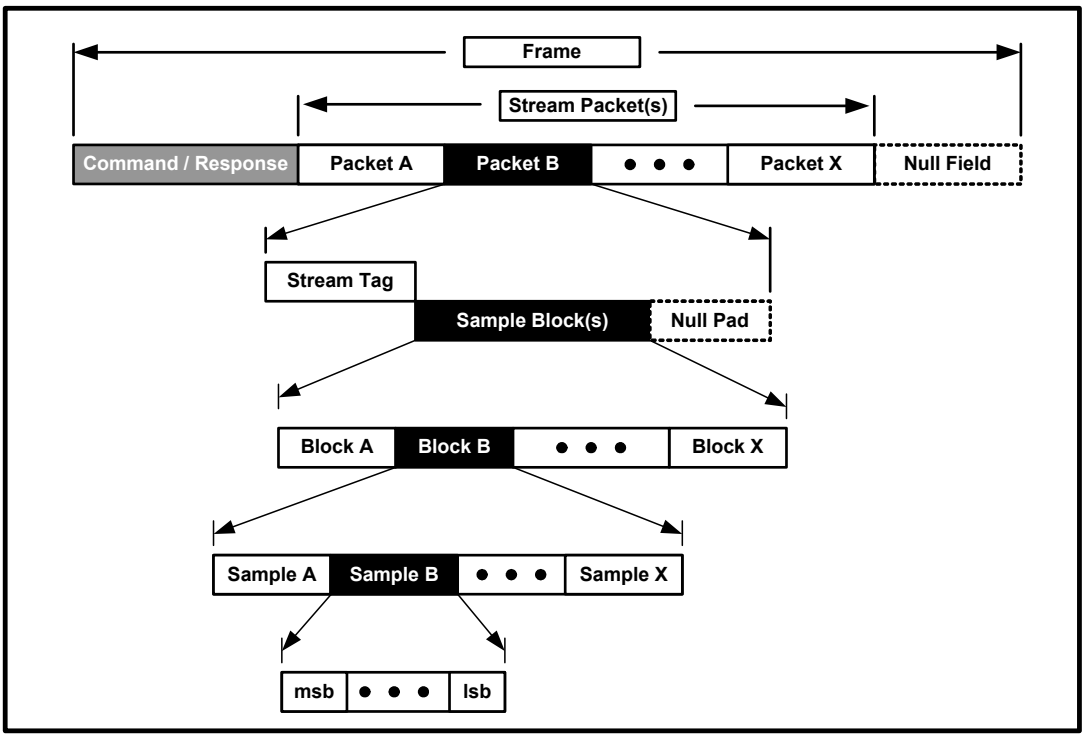

Figure 21. Frame Composition

Command/Response Field: Used for link and codec management. One of these fields appears exactly once per frame, most significant bit first, and is always the first field in the frame. It is composed of a 40-bit Command Field on each outbound frame and a 36-bit Response Field on each inbound frame. The primary component of each of these fields is the 32-bit Verb/Response structure. The remaining eight command bits in the outbound frame, and four response bits in the inbound frame, are either reserved or special purpose bits (see Sections 7.3 and 7.3.1)

Stream Packet: The logical "envelop" in which data are transferred on the link, consists of one *stream tag* plus one or more *sample block(s)*, the number of which are specified by the "Block Multiple" field of the Stream Descriptor Format registers. (See Section 3.3.34 for more information on the Stream Descriptor Format registers.) In addition, an inbound stream packet may include 4 bits of 0 padding (see Section 5.3.3.2) at the end of the packet only. No padding is allowed between stream packets nor between the Command/Response Field and the first stream packet (if there is one). With the exception of source synchronous input streams (see Section 5.4.3), no stream is allowed to generate more than one stream packet per frame.

Stream Tag: The label at the beginning of each stream packet that provides the associated stream ID. All data in one stream packet belongs to a single stream. The Tag format and method of transmission differ for inbound and outbound streams.

Sample Block: A set of one or more *samples*, the number of which is specified by the "Channels" field of the Stream Descriptor Format registers (see Section 3.3.34). Samples in a given sample block are associated with a single given stream, have the same length or sample size, and have the same time reference or sample point.

Furthermore, samples within a block are contiguous – no padding is permitted between samples. Ordering of samples within a block is always the same for all blocks in a given stream. The binding between logical channel assignment (e.g., left, right, center) and sample order will always be known to the device driver and may also be known to the codec (see Section 7.3.3.11) but is irrelevant to the controller.

Sample: A set of bits providing a single sample point of a single analog waveform, with length specified by the "Bits per Sample" field of the Stream Descriptor Format registers. Samples are always transmitted MSB first on the link.

Null Field: The remainder of the bits contained in each inbound or outbound frame that are not used for Command/Response, or for packets, are a null field. A null field must be transmitted as logical 0's. Codecs and controllers are required to always transmit all data for a given frame contiguously starting with Command/Response followed by all stream packets to be sent in that frame. Null fields may only be sent after all Command /Response and stream packet(s) have been sent within a given frame. Null fields between stream packets, or Command /Response fields and stream packets, are prohibited.

5.3.2 Output Frame Formatting

5.3.2.1 Outbound Stream Tags

Outbound Stream Tags are 8 bits in length and are transmitted at a double pumped rate as side band information on **SYNC**. Outbound stream tags delineate the beginning of each new outbound stream packet. Outbound stream tags are transmitted on **SYNC** so as to align with the last eight (data) bits of the preceding stream packet or Command Field.

The format of an outbound tag is shown in Figure 22. It is composed of a four bit preamble which is signaled as three **SDO** bit times high followed by one **SDO** bit time low. This is immediately followed by a 4-bit Stream ID, most significant bit justified, which identifies the specific stream to which the subsequent sample blocks are associated. Sample blocks are transmitted on **SDO** immediately following the least significant bit of the outbound tag, as shown in Figure 22.

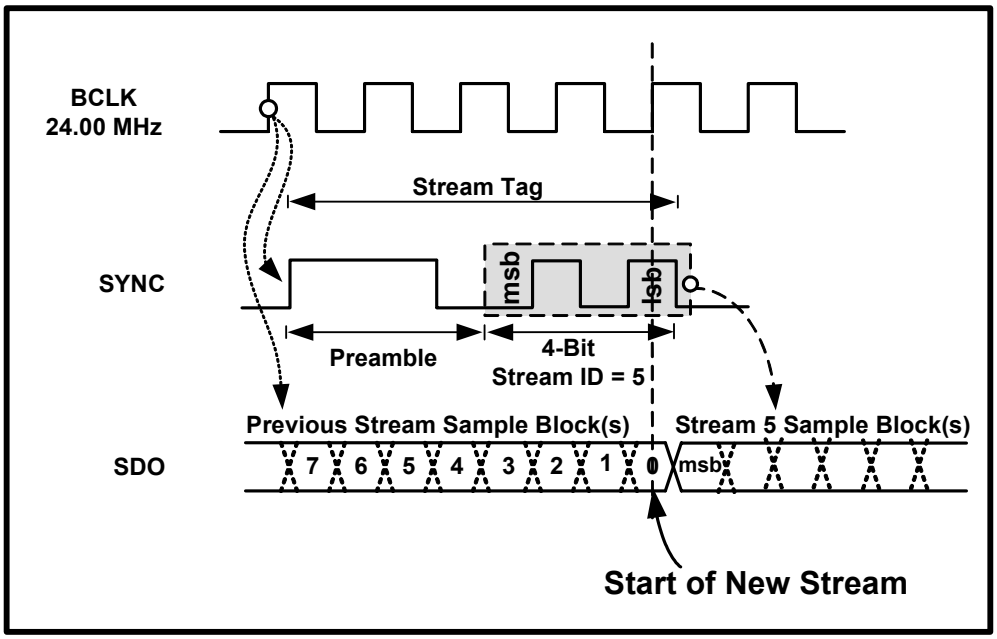

Figure 22. Outbound Stream Tag Format and Transmission

5.3.2.2 Outbound Frame Overview – Single SDO

Figure 23 shows a typical outbound frame. Outbound frames start and end between the falling edges of successive Frame Syncs. The first 40 bits of an outbound frame are dedicated for the Command field and are used to send commands to codecs. The Controller transmits the tag for the first outbound packet on **SYNC** during the last eight bit times (four **BCLK** cycles) of the Command field. The sample block(s) for the first packet is transmitted on **SDO** immediately following the Command field. The controller transmits stream packets within the frame in a contiguous manner according to the Format specification of all active Output Stream Descriptors (Section 3.3.34) until all packets for that frame have been transmitted; controllers are prohibited from transmitting null fields between outbound stream packets. There is no proscribed order in which the different stream packets are to be transmitted. Controllers are required to transmit a null field for the remaining bits within an outbound frame when the transmission of the stream packets completes before the end of the frame. Null fields are only permitted after all outbound stream packets have been transmitted.

In the event that software errantly over-subscribes the outbound link (**SDO**), and all the proscribed data would overrun the frame, then data transmission for that frame shall be terminated at the low-going edge of the Frame Sync marker, and a new frame shall be started with all Command field bits in the proper place. The sample(s) that were dropped in this frame termination shall not be re-transmitted in the next frame, and the further behavior of the associated stream(s) is undefined.

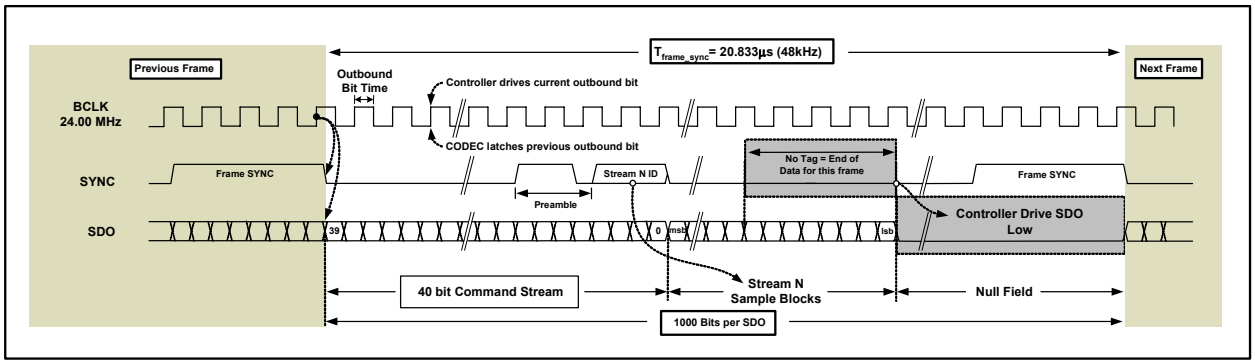

Figure 23. Outbound Frame With Null Field

5.3.2.3 Outbound Frame Overview – Multiple SDO

Each Output Stream Descriptor Control register (see Section 3.3.34) contains a "Stripe Control" field indicating whether this stream should be transmitted on multiple **SDO** signals or only on **SDO**$_0$. When software initializes the stream, it determines multiple **SDO** capability of the target codec(s) as well as the controller and sets this control accordingly. When a stream is to be transmitted on multiple **SDO**s, it proceeds as above, except the data is "bit-striped" – sent every other bit on alternating **SDO**s – thus completing the transmission in less time, and freeing **SDO**$_0$ for other data. When striping, the first bit (MSB) is transmitted on **SDO**$_0$, the second bit on **SDO**$_1$, and so forth as show in Figure 24. **SDO**$_0$ is used again when all **SDO**s assigned to this stream have transmitted a bit. When a stream is to be transmitted only on **SDO**$_0$, the higher order **SDO**s transmit logical 0's.

The Command Field is always transmitted on **SDO**$_0$ without striping, but the controller copies the Command Field on all higher order **SDO**s in order to ensure that they toggle. This allows multi-**SDO** codecs to determine and report whether all **SDO** connections are present once normal like operation begins after codec initialization.

When an outbound stream is common to multiple codecs, the stream must always be transmitted on the lowest order **SDO**s common to the codecs in question. Software is responsible for configuring the controller DMA engine(s) and codec resources appropriately.

In general, striping reduces the total number of outbound bit times that a given stream's sample blocks occupy in a given frame. As an example, a 16-bit mono stream occupies 16 outbound bit times if it is transmitted on a single **SDO**. The same stream will occupy eight outbound bit times if it is transmitted on two **SDO**s and four outbound bit times if it is transmitted on four **SDO**s. Note that the four **SDO** case breaks the ability to transmit the stream tag for a succeeding outbound stream since eight outbound bit times are required to transmit the stream tag in question. Given this problem, software is required to adhere to the following two equations when determining how many **SDO**s to stripe across for a given stream in a multi-**SDO** system.

For sample rates greater than 48 kHz:

$$\left\lceil \frac{(number\ \_of\ \_channels\ ) \times (bits\ \_per\ \_sample\ ) \times \left(\frac{Sample\ \_Rate}{48000\ Hz} \right)}{number\ \_of\ \_SDOs} \right\rceil \geq 8$$

For samples rates less than or equal to 48 kHz:

$$\left\lceil \frac{(number\_of\_channels) \times (bits\_per\_sample)}{number\_of\_SDOs} \right\rceil \geq 8$$

This will ensure that all striped stream occupy the minimum 8 outbound bit times such that stream tags can be transmitted.

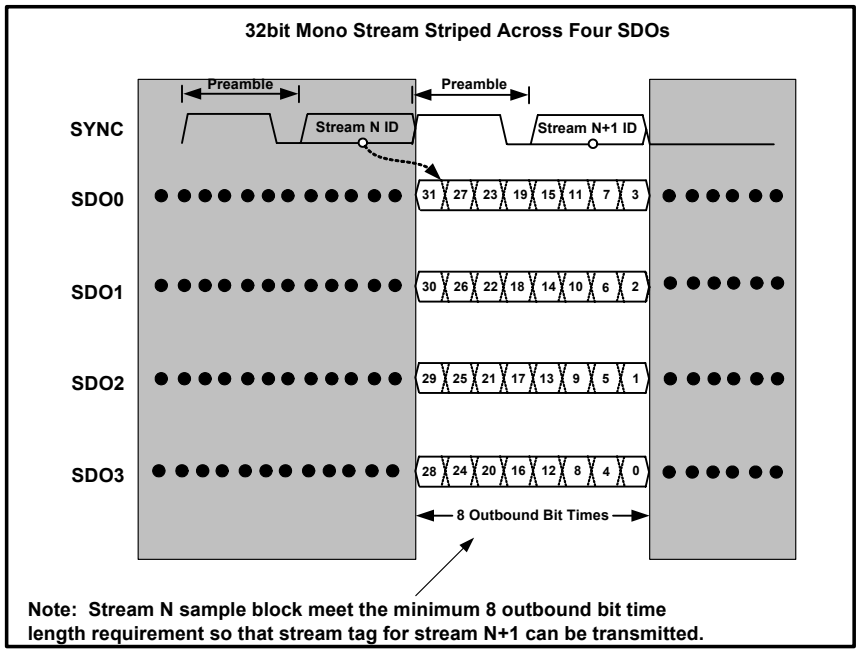

Figure 24. Outbound Striping Example

5.3.3 Input Frame Formatting

5.3.3.1 Inbound Stream Tags

An Inbound Stream Tag is 10 bits in length, and is transmitted "in-line" at a single pumped rate on **SDI**, immediately preceding the associated inbound sample block(s). The format of an inbound tag is shown in Figure 25. It is composed of a 4-bit stream ID, which identifies the specific stream to which the sample block(s) belong, followed by a 6-bit data length field that provides the length, in bytes, of all sample blocks within the given stream packet. Stream ID and Data Length are transmitted most significant bit justified. The associated sample blocks are transmitted on **SDI**

immediately following the least significant bit of the inbound stream tag. Stream tags immediately follow, without padding, the Response Field or prior stream packet.

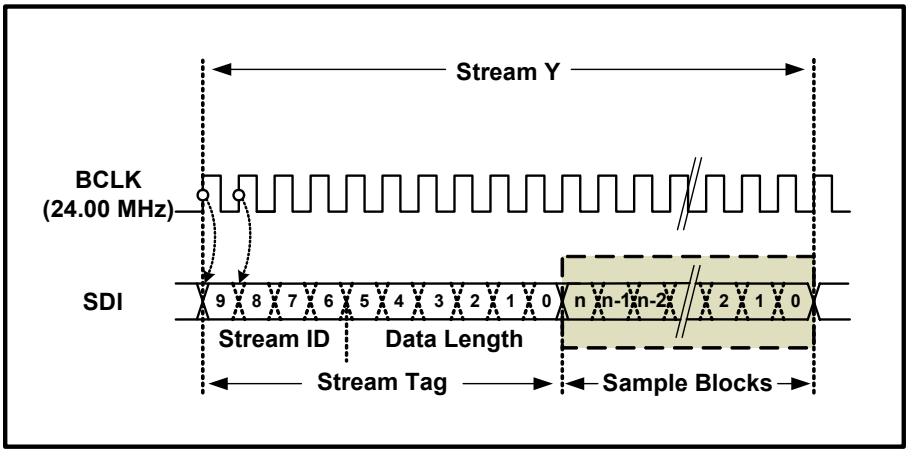

Figure 25. Inbound Tag Format and Transmission

5.3.3.2 Zero Padding Inbound Stream Packets

Whereas outbound stream packets may have arbitrary bit length, inbound stream packets differ in that the total length of the contiguous sample blocks within a given stream packet must be of integral byte length. Since all defined sample sizes are not of integral byte length (e.g., 20-bit samples), codecs are required to pad 0's at the end of any inbound stream packet in which the contiguous sample data does not conclude on a byte boundary. Thus, the Data Length field of the inbound stream tag identifies the exact position within the serial input data where the subsequent stream tag will be found.

As an example, consider a 48-kHz mono stream using 20-bit samples. Since in this case there would be only one sample in the stream packet, and since 20 bits is not a valid inbound stream packet length, the codec must add four 0's (pad) at the end of the packet to make the sample block an integral byte length, and the length field in the stream tag preceding this stream packet must be set to 3 (bytes). This is illustrated in Figure 26.

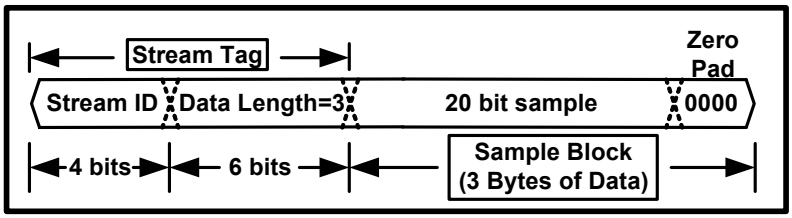

Figure 26. 20-bit Mono Stream With Data Padded to Nearest Byte

Note that had the example been a 48-kHz stereo stream or 96-kHz mono stream using 20-bit samples, the stream packet would have contained two samples or 40 bits, which is an integral byte length, and no padding would have occurred. In no case will padding length exceed 4 bits, and padding only occurs at the end of sample transmission when a stream packet is composed of an odd number of 20-bit samples.

5.3.3.3 Inbound Frame Overview

Figure 27 shows a typical inbound frame. Inbound frames start and end between the falling edges of successive Frame Syncs. The first 36 bits of an inbound frame are dedicated for the Response Field, which codecs use for sending responses to controller commands. The Codec transmits the first stream packet on **SDI** immediately following the Response Field. The codec transmits inbound stream packets in a contiguous manner until all packets for that frame have been transmitted. There is no proscribed order in which the different inbound stream packets are to be transmitted. Codecs are prohibited from transmitting null fields between inbound packets.

A stream tag indicating a stream packet length of zero must immediately follow the last stream packet to be transmitted. Such a stream tag marks the completion of data transmission within that frame, and the remaining valid bit positions are set to the null field. In the event there are less than 10 valid bit positions remaining in the frame after the last stream packet, then no termination tag is transmitted, and the remaining bits are the null field. Note that it is permissible to specify stream zero in this termination tag making the tag 10 logical 0's – thereby appearing to be part of the null field.

If a codec is ever commanded by software to send more data in one frame than can legally fit into a single frame, then data transmission for that frame shall be terminated at the low-going edge of the Frame Sync marker, and a new frame shall be started with all Response bits in the proper place. Data that was dropped in this frame termination shall not be re-transmitted in the next frame, and the further behavior of the associated stream(s) is undefined.

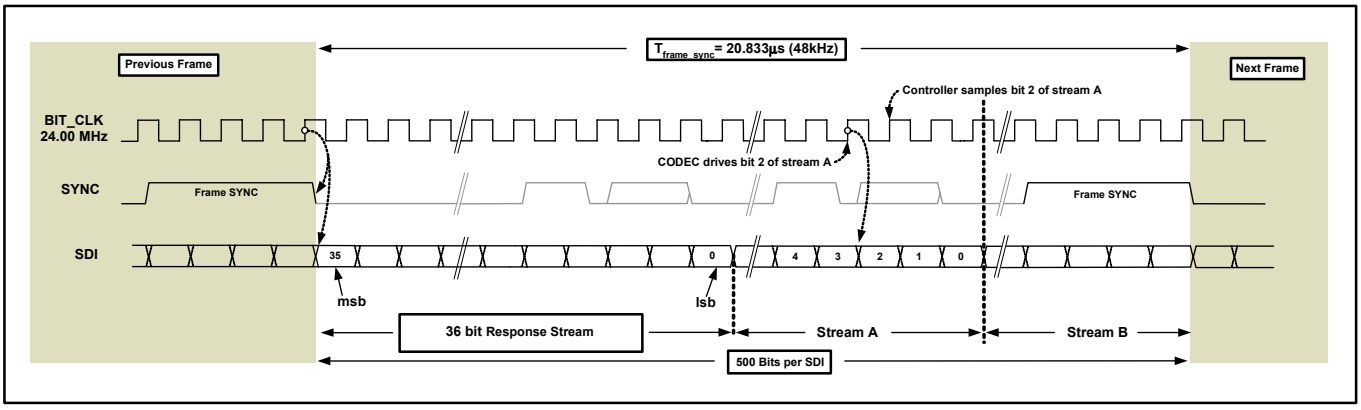

Figure 27. Inbound Frame With No Null Field

5.3.3.4 Stream Transmission Over Multiple SDI Signals

Some multi-function codecs may require multiple **SDI** connections to achieve the bandwidth needed to function completely. In that case, codecs may send data over two or more **SDI** signals, each of which operates independently. Logically, this appears to the controller as multiple codecs. See Section 7.3.2 for more information on Multi-**SDI** codecs.

In the case that a single virtual inbound stream exceeds the data transfer limits of a single **SDI** signal, software must divide that stream into multiple real streams – each with separate stream IDs, transmitting on separate **SDI** signals – according to the capabilities of the subject codec. This

implies that the virtual stream will also be divided into multiple separate memory input buffers, which software is responsible to appropriately combine.

5.4 Handling Stream Independent Sample Rates

The Link is optimized for sample rates of 48 kHz and integral multiples thereof. Transmitting one sample per channel in each frame results in a sample delivery rate of 48 kHz; two samples per frame results in a 96-kHz delivery rate, etc. Notwithstanding this optimization, the High Definition Audio Specification defines the means for delivering samples at all common alternative sample rates as shown in Table 56. Furthermore, each stream defines its own sample rate independent of any other stream.

Since the Link is source synchronous and has no codec initiated flow control (i.e., it is purely isochronous), the controller generates all sample transfer timing. This implies that the controller and link must also provide for an *exact* sample-rendering rate at the codec, in order to avoid any long term drift between sample delivery and rendering and the subsequent over(under)-run of codec buffers. Both sample delivery and rendering timings are derived from the Link **BCLK** as defined below.

5.4.1 Codec Sample Rendering Timing

All common sample rates are integral multiples (or submultiples) of one of two standard base rates[1] both of which must be derived from the Link **BCLK**. All codecs must derive the base rate of 48.0 kHz as **BCLK** divided by 500 or directly from the link framing **SYNC** signal. Multiples (e.g., 96.0 kHz, 192.0 kHz) and submultiples (e.g., 16 kHz or 8 kHz) of this base rate must be derived as the appropriately divided **BCLK**.

If a codec supports sample rates using the base rate of 44.1 kHz, then this base rate must be derived as an exact 147/160 mathematical multiple of the 48.0-kHz base rate or as an exact 147/80,000 mathematical multiple of **BCLK**. Again, multiples or submultiples of this base rate must be derived as the appropriately divided **BCLK**.

All codec render/sample timing must be an appropriate multiple or submultiple of one of the two base rates as described above. Table 56 itemizes all the sample rates defined in (but not necessarily required by) in the High Definition Audio Specification together with their multiple (submultiple) and base rate from which each must be derived.

[1] Source synchronous digital audio (e.g., S/PDIF) and modem *input* are exceptions to this rule. In this case, sample timing is defined by the received stream, and not by the link **BCLK**. See Section 5.4.3 for further explanation.

Table 56. Defined Sample Rates

	Base Rate (kHz)	
Multiple	**48.0**	**44.1**
1/6	8.0	
1/4		11.025
1/3	16.0	
1/2		22.05
2/3	32.0	
1	48.0	44.1
2	96.0	88.2
4	192.0	176.4

5.4.2 Link Sample Delivery Timing

Having defined exact rendering/sampling clocks, the task remains to provide for sample delivery across the Link, without incurring any long term drift that would overrun or underrun codec buffers. This must be accomplished under the constraints of a fixed frame rate on the link (48 kHz) and no codec-directed flow control.

48-kHz Delivery Timings

This becomes straight forward for streams whose sample rate is a natural harmonic of 48 kHz. In this case the base rate multiple also defines the number of complete sample blocks that must be transmitted in each frame. For either input or output, there must be transmitted on the link "*y*" complete sample blocks (block includes one sample for each channel in the stream) on every n^{th} frame as shown in Table 57. For example, with a multiple of 4 (192 kHz sample rate), there must be four sample blocks transmitted in every frame on the link; with a submultiple of 1/6 (8-kHz sample rate), there must be one sample block transmitted every one in six frames on the link – the intervening five frames will contain no sample for this stream.

For streams requiring multiple samples in a given frame, those samples will be transmitted contiguously within a single stream packet; multiple samples from the same stream will not be spaced out within the frame. When a stream is started which does not transmit sample blocks in every frame, the first full frame occurring after the stream is started always contains a sample block followed by the requisite number of frames without samples. If such a stream were subsequently stopped and restarted, the first occurring full frame would again contain a sample block regardless of the frame sequence in which that stream had been stopped.

Table 57. Sample Transmission Rate

Multiple	*y* number of blocks	*n* Frame Frequency
1/6	1	6
1/4	1	4
1/3	1	3
1/2	1	2
2/3	2[2]	3
1	1	1
2	2	1
4	4	1

44.1-kHz Delivery Timings

Since the link frame rate is fixed at 48 kHz, streams using a base rate of 44.1 kHz must have samples transmitted on a cadence creating the slightly lower aggregate transmission rate to match the slightly lower rendering rate. For streams running at a sample rate of 44.1 kHz, instead of transmitting one sample block in each frame (as happens for 48.0 kHz rate streams), there are occasional frames that will not contain a sample generating the following cadence.

12-11-11-12-11-11-12-11-11-12-11-11-11- (repeat)

In this cadence, the dashes indicate frames that do not contain a sample block. Therefore, there are 12 frames containing one sample block each, followed by one frame with no sample block, followed by 11 frames with one sample block each, etc. Following the empty frame at the end, the cadence repeats continuously generating exactly 147 sample blocks every 160 frames and avoiding any long-term drift between sample delivery and rendering clock.

For streams running at multiples or submultiples of 44.1 kHz, this cadence is applied using the frame frequency rules from Table 57 as follows. Frames at a given (sub)multiple that would not have contained a sample block will also not contain a sample block after applying the "12-11" cadence. However, frames at the same (sub)multiple that would have contained a sample block absent the "12-11" cadence will now apply the cadence as a further criteria, and the sample block is only transmitted when the cadence so indicates. In other words, the cadence will be applied and incremented only on those frames that would otherwise have contained a sample block.

To illustrate, a sample rate of 22.05 kHz has a resulting transmission pattern of:

1*1*1*1*1*1*1*1*1*1*1*1* –* 1*1*1*1*1*1*1*1*1*1*1*1* –* 1*1*1. . . , etc.

In this case, the asterisks represent a frame in which there was no sample block because Table 57 only called for a sample block on every other frame. Additionally, the dash represents a frame without data because the cadence stepping called for an empty frame in that position. Note that the

[2] Note that the 2/3 multiple of the base frequency sends two samples every third frame rather than sending one sample in each of two out of three frames. This architectural and implementation optimization does have the side effect of doubling the required codec buffering for codecs supporting related sample rates; however, this should not be a significant burden since the samples associated with this 32.0-kHz rate are likely to be small.

combined application of these two rules creates three empty frames occasionally. Similarly, a sample rate of 11.025 kHz has a resulting transmission pattern of:

1***1***1***1***1***1***1***1***1***1***1*** –*** 1***1***1. . . , etc.

Sample rates that are integral multiples of 44.1 kHz (88.2 kHz and 176.4 kHz) apply the "12-11" cadence rule just as a 44.1-kHz sample rate would, except that non-empty frames contain multiple (two or four) sample blocks instead of just one, again per rules of Table 57. When an output stream that uses the "12-11" delivery cadence is started, the cadence sequence is always started at the beginning (12), and the first full frame occurring after the stream is started always contains a sample block, followed by the requisite number of frames without samples. If such a stream were subsequently stopped and restarted, the cadence must also be restarted at the beginning, and the first occurring full frame must again contain a sample block regardless of the frame sequence in which that stream had been stopped.

These framing sequences must be strictly adhered to for all outbound (**SDO**) data transmitted by the Controller. Inbound (**SDI**) data transmitted by the codec should generally follow the same rules, however, a codec is permitted to deviate insofar as this more optimally spaces samples for minimizing codec buffer management. A codec is never allowed to transmit more sample blocks than indicated by Table 57 in a single frame.

5.4.3 Source Synchronous Input

For certain input streams such as S/PDIF audio or modems, sample timing is defined not by **BCLK** but by the stream source. In these cases, the codec has some latitude in transferring data on the Link. In such cases, software initializes the stream by setting the Block Multiple field to correspond to the nominal rate[3] of the source synchronous input stream. This setting occurs both in the Descriptor Format Register of the controller as well as in the codec. The codec is ensured that particular size of data transmission in each and every frame but may not utilize all this allocated bandwidth because the actual transfer rate is slightly lower. The codec may also have occasional need of an extra data transmission if the actual transfer rate is slightly higher. In this case, the codec may insert one or more additional complete sample blocks on the link, either within its normal allocated stream packet (making it larger than the nominally allocated size), or by creating a second stream packet within the frame. Either way, the codec must not add this extra block(s) unless it first determines that all allocated blocks in all streams either have been or can be transmitted within the frame and that there is ample space left[4] in the frame for the extra block(s). If not, the codec must wait until a subsequent frame to send the extra block(s). In no case may the codec transmit extra data less than one full sample block in size, as incomplete blocks will be dropped by the controller.

[3] This nominal rate can always use the 48-kHz base rate, since from a link allocation point of view, it is the same as 44.1 kHz.

[4] Ample space to transfer an extra sample block may be ensured by design in some low bandwidth codecs or modems. In this case, software allocating the input bandwidth could over-allocate, thus obviating the need for specific codec logic to perform an available space calculation.

5.5 Reset and Initialization

In order to minimize link housekeeping sequences, the High Definition Audio Link has been designed with a single initialization sequence that always follows a reset sequence. This section describes all of these, beginning with the two types of reset that occur within an High Definition Audio system.

- Link Reset – generated by assertion of the link **RST#** signal affecting all Link interface logic in both codec and controller.
- Codec Function Group Reset – generated by software directing a reset command (verb) to a specific function group within the codec affecting only the targeted codec and nothing else on the Link.

5.5.1 Link Reset

A link reset is signaled on the Link by assertion of the **RST#** signal. (See Figure 28 and Figure 29 for specific **RST#** timing.) Link reset results in all Link interface logic in both codec and controller, including registers, being initialized to their default state. Note, however, that codec functional units may contain logic that is not reset with the **RST#** signal including logic associated with power management functions, such as power state information or Caller ID in a modem codec. Codec functional logic in general must be reset by a soft command or verb; see Section 7.3.3.33, for more information.

The link-reset sequence occurs in response to three classes of events:

1. Reset occurring for any reason on the Controller's host bus (e.g., PCI), including system power sequencing. This results in the asynchronous assertion of **RST#** on the Link.
2. Software initiating link reset via the CRST# bit in the Global Control Register (see Section 3.3.7). This results in the synchronous assertion of **RST#** on the Link per the Link Reset Entry Sequence.
3. Certain software initiated power management sequences (see Section 5.6).

The controller drives all **SDO** and **SYNC** outputs low when entering or exiting link reset. While the link in is reset, with **RST#** asserted, and the controller is powered or in a wake enabled state, codecs drive all **SDI** signals low, and controllers, at a minimum, must pull all **SDO, SYNC**, and **BCLK** outputs low. Codecs may drive **SDI**s high to signal a power state change request as specified in Section 5.6. Codecs that support test mode must float their **SDI** in reset and may only drive **SDI** high to signal a power state change request.

5.5.1.1 Entering Link Reset

A controller may only initiate the link reset entry sequence after completing any currently pending initialization or state change requests (see Section 5.5.3). Figure 28 shows the required sequence when entering link reset, specifically:

1. The controller synchronously completes the current frame but does not signal Frame Sync during the last eight **SDO** bit times.
2. The controller synchronously asserts **RST#** four (or more) **BCLK** cycles after the completion of the current frame.
3. **BCLK** is stopped a minimum of four clocks, four rising edges, after the assertion of **RST#**. Note that **BCLK** must be stopped in a glitch-free manner only when it is in the low state.

In the event of a host bus reset, the above sequence does not complete, and **RST#** is asynchronously asserted immediately and unconditionally.

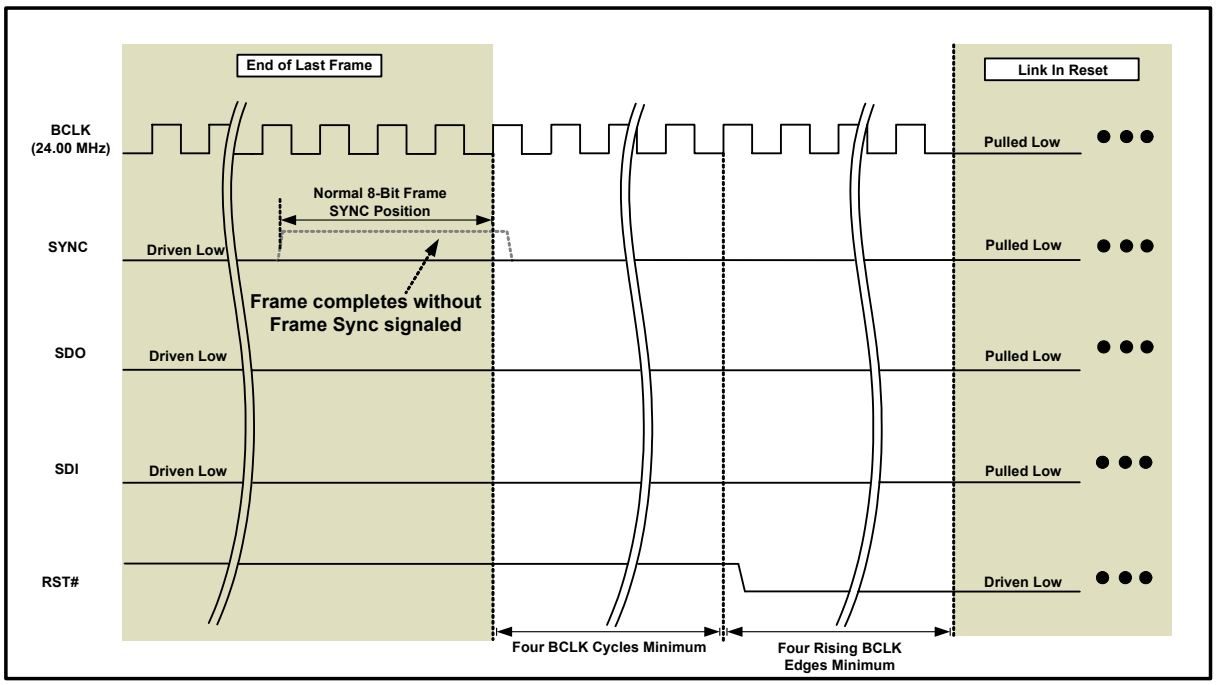

Figure 28. Link Reset Entry Sequence

5.5.1.2 Exiting Link Reset

Regardless of the reason for entering Link Reset, it may be exited only under software control (see the description of the CRST# bit in the Global Control Register in Section 3.3.7) and in a synchronous manner as shown in Figure 29. This sequence is as follows:

1. The controller provides a properly running **BCLK** for a minimum of 100 µs, 2400 **BCLK** cycles or more before the de-assertion of **RST#**. This allows time for codec PLLs to lock. **SYNC** and all **SDO** signals must be driven low concurrently with the **BCLK** startup.
2. The **RST#** link signal is de-asserted.
3. The **SYNC** commences signaling valid frames on the link with the first Frame Sync occurring a minimum of four **BCLK** cycles after the de-assertion of **RST#**.
4. Codecs must signal an initialization request, via **SDI**, within the first 25 Frame Syncs relative to the de-assertion of their respective **RST#** signal. See Section 5.5.3 for information on codec initialization.

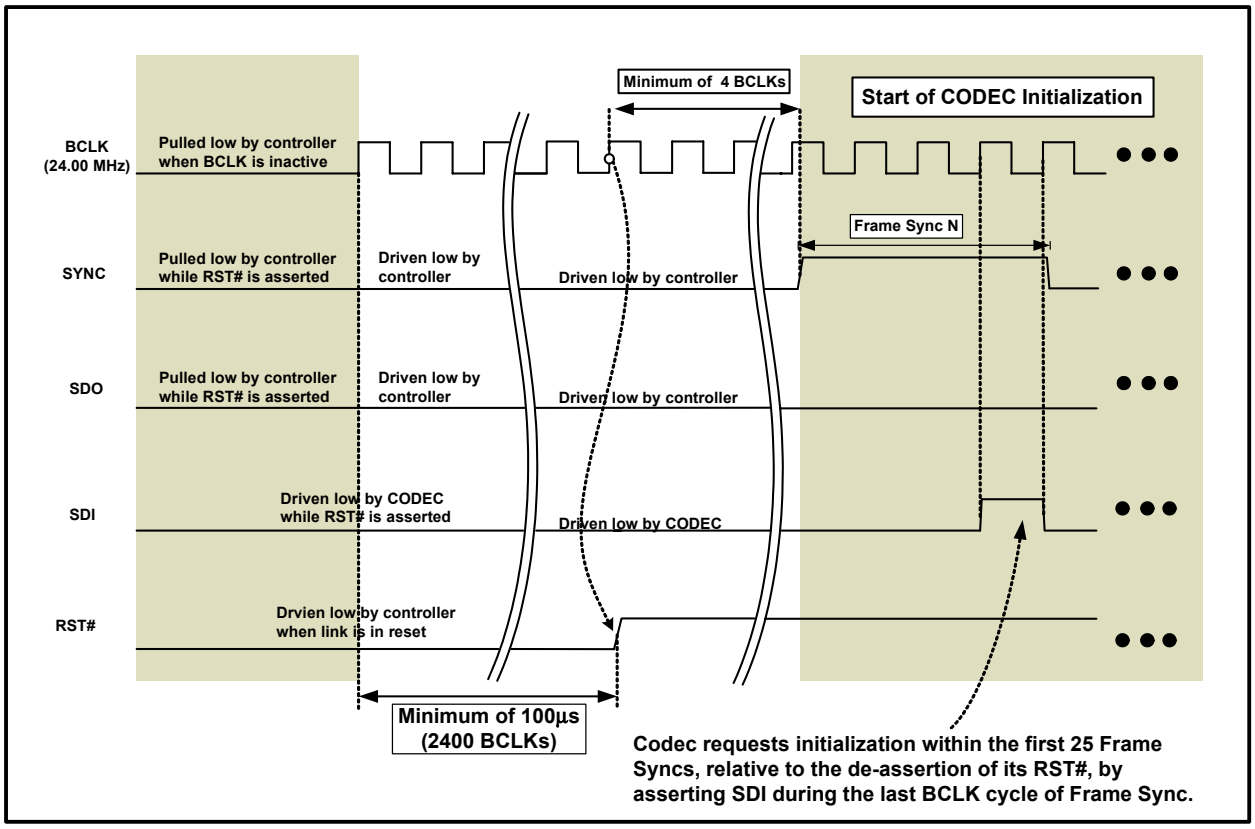

Figure 29. Link Reset Exit Sequence

5.5.2 Codec Function Group Reset

A codec function group reset allows software to initialize/reset a specific Codec function group without affecting or interrupting the operation of the Link. A codec function group reset is initiated via the Function_RESET command verb, as described in Section 7.3.3.33, and results in all logic within the targeted function group being driven to its default or reset state.

5.5.3 Codec Initialization

Immediately following the completion of any reset sequence, all affected Codecs proceed through a codec initialization sequence as described in this section and shown in Figure 30. The purpose of this initialization sequence is to provide each codec with a unique address by which it can thereafter be referenced with Commands on the **SDO** (broadcast) signal. During this sequence, the Controller provides each requesting codec with a unique address using its attached **SDI** signal(s). Controllers are required to support independent (simultaneous) initialization on all **SDI** signals. Independent initialization allows for codecs to be connected to the interface, be hard reset, and assigned an address even when the link is in normal running state which is required for hot docking.

The codec initialization sequence occurs across three contiguous frames immediately following any reset sequence. During these three frames, codecs are required to ignore all outbound traffic present on **SYNC** (stream tags) and **SDO** (commands and stream packets). These three frames,

labeled the “Connect Frame”, the “Turnaround Frame”, and the “Address Frame”, are described sequentially below.

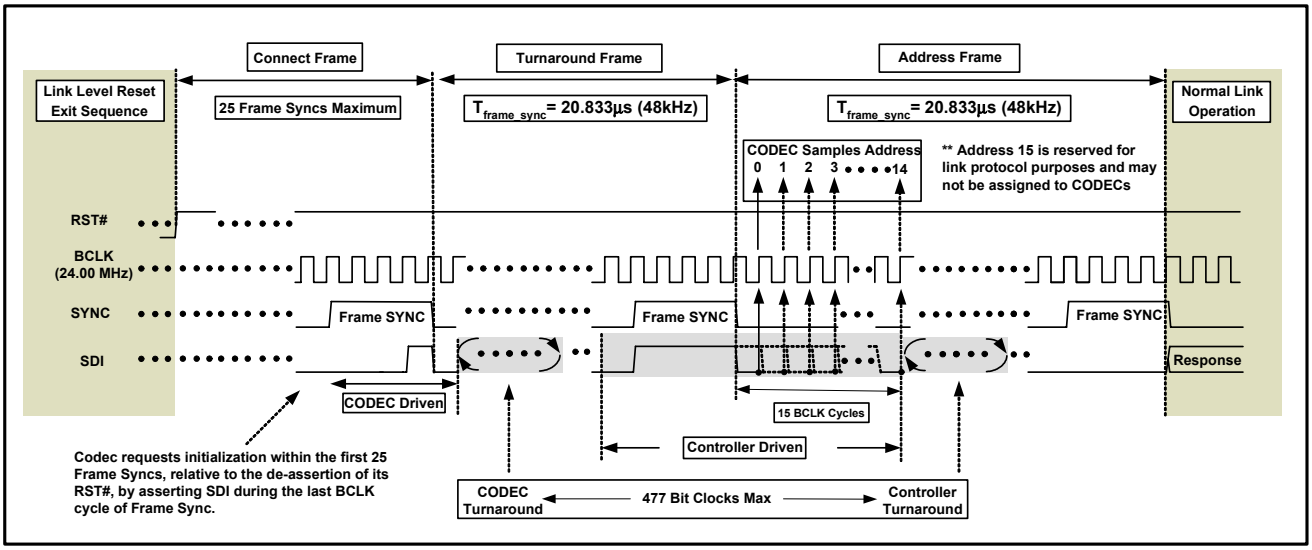

Figure 30. Codec Initialization Sequence

5.5.3.1 Connect and Turnaround Frames

In the codec Connect and Turnaround Frames, shown in Figure 31, the codec signals its request for initialization on **SDI** and then releases **SDI** (turnaround) to be driven by the controller in the subsequent address frame.

The Frame Sync marking the end of the Connect Frame and the beginning of the Turnaround Frame is the same one as one of the following:

- The “first Frame Sync” in which the codec signals an initialization request when coming out of a link reset state (described in Section 5.5.1.2).
- Any Frame Sync in a normally running system that is coincident with a codec initialization request.

The codec may distinguish a Frame Sync marker from an outbound stream tag on **SYNC** by detecting **SYNC** driven high for four consecutive **SDO** bit times. Once Frame Sync has been detected, and PLL lock has been achieved, codecs signal an initialization request by synchronously driving **SDI** high during last bit clock cycle of Frame Sync (see Section 5.5.1.2 for codec initialization request requirements). **SDI** must be asserted for the entire **BCLK** cycle and must be synchronously de-asserted on the same rising edge of **BCLK** as the de-assertion of the Frame Sync. If a codec is requesting a power state change when the Link is in a low power state (**BCLK** and **SYNC** not operating), then it asynchronously drives **SDI** high continuously until it detects the de-assertion of **RST#**. It must then asynchronously drive **SDI** low with the de-assertion of the **RST#**. Codecs are only permitted to signal an initialization request on a null input frame, a frame in which no response stream or input streams are being sent.

A codec that may be put into a “hot plug” or “hot dock” situation has the additional issue that its link isolation circuit may time out asynchronously with respect to the already running Link thus

causing its **RST#** to be de-asserted asynchronously and at an unknown time. Codecs are required to verify that **SYNC** is sampled low for two **BCLK** cycles, four **SDO**/**SYNC** bit times, before starting scanning for the first four high bits of Frame Sync. This ensures that codecs do not request initialization during transmission of outbound stream tags. In the Turnaround Frame, codecs and controllers are required to turn **SDI** around (reverse driving direction) upon the completion of the Connect Frame. To do this, the codec actively drives **SDI** low for one **BCLK** cycle immediately following the de-assertion of **SYNC** at the end of the Connect Frame. The codec then puts its **SDI** drivers in a high impedance state at the end of the first **BCLK** cycle in the Turnaround Frame. Four **BCLK** cycles before the end of the Turnaround Frame, **SYNC** and all **SDI**s that have signaled an initialization request are driven high by the controller. These **SDI** signals remain driven high through the end of the Turnaround Frame in preparation for the subsequent address frame.

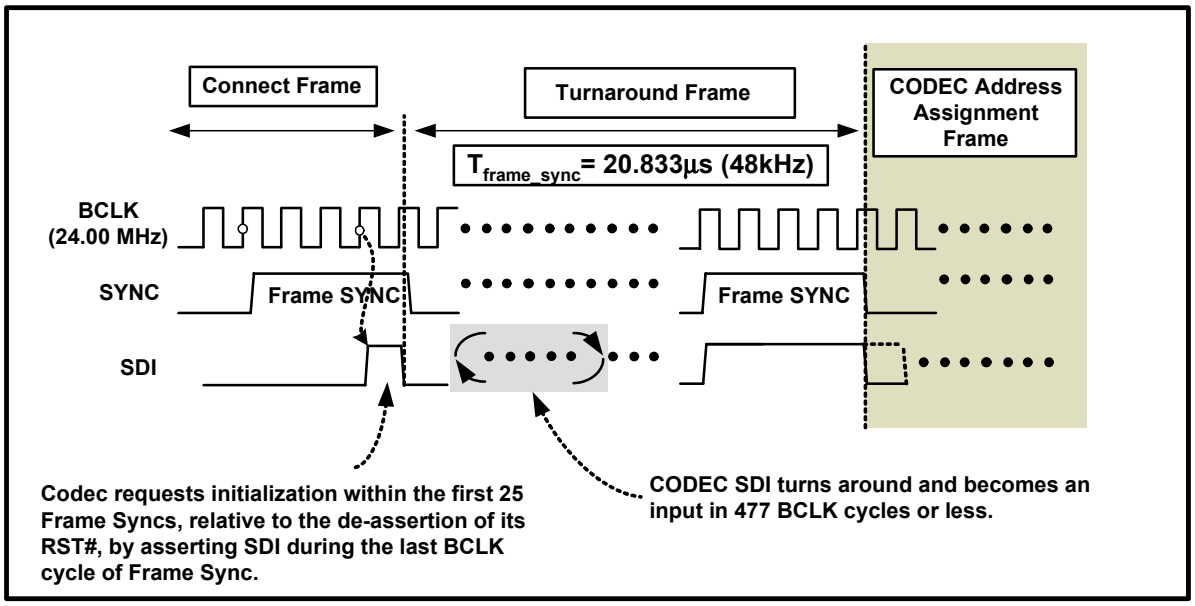

Figure 31. Codec Connect and Turnaround Frames

5.5.3.2 Address Frame

Figure 32 shows the codec address frame consisting of an address assignment followed by a final **SDI** turnaround in preparation for normal link operation. During the address frame, **SDI** is a codec input and is driven by the controller beginning in the last four **BCLK** periods (Frame Sync) of the Turnaround Frame. The falling edge of Frame Sync marks the start of codec address assignment. Address assignment is indicated by the controller holding each **SDI** high for the number of **BCLK** cycles equal to the numeric ID of that particular **SDI**; i.e., $\mathbf{SDI}_0$ is held high for zero **BCLK** cycles after the beginning of the frame, and $\mathbf{SDI}_n$ is held high for *n* **BCLK** cycles. Thus the unique address of the codec becomes the ID of its attached **SDI**.

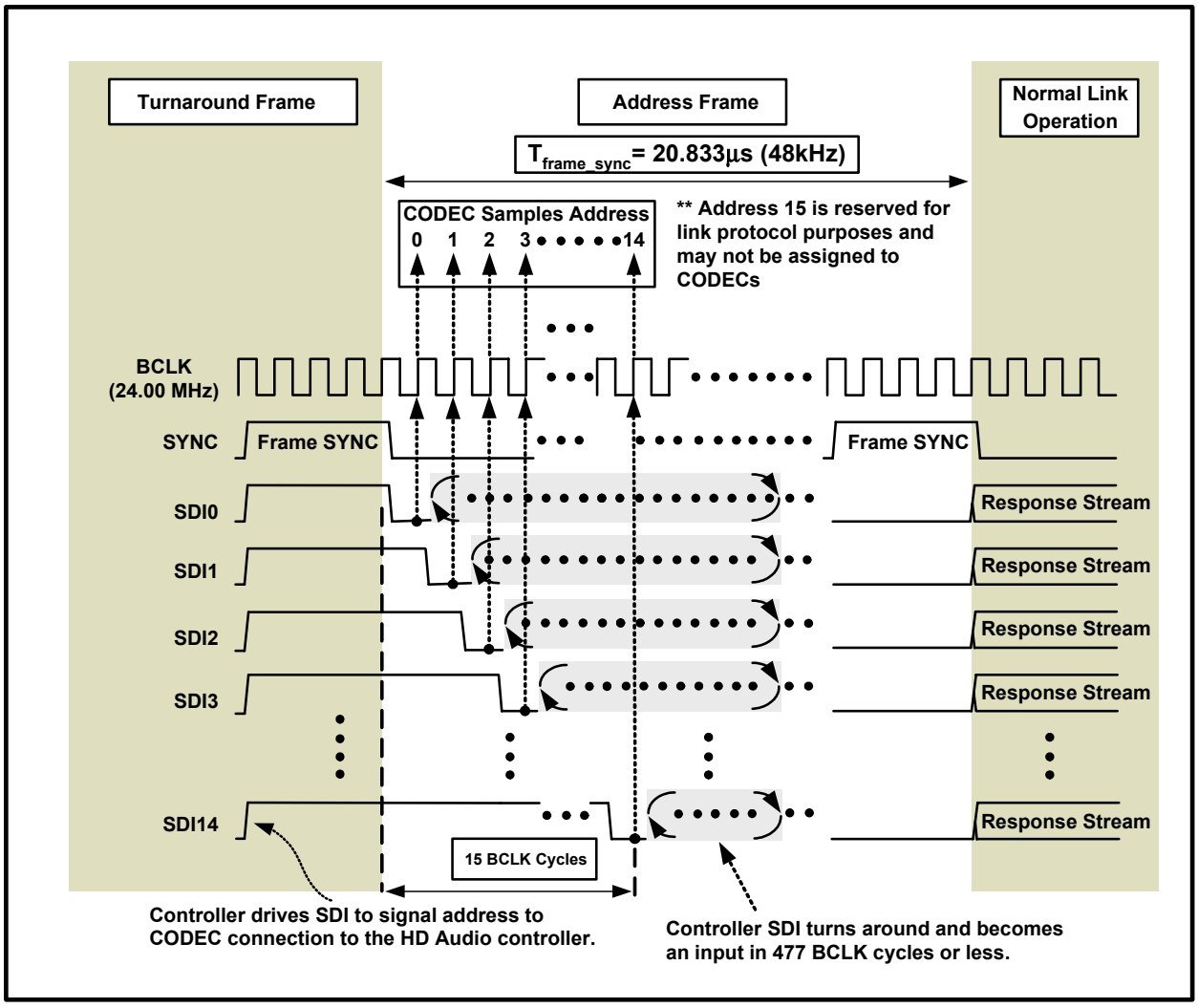

Figure 32. Codec Address Assignment Frame

Codecs count from zero to fourteen starting on the rising edge of **BCLK** following the de-assertion of Frame Sync, and sample the value of this count for their unique address on the first rising edge of **BCLK** in which **SYNC** and **SDI** are both sampled low.

The controller must put its **SDI** drivers in a high impedance state by the rising edge of the 18th **BCLK** of the address frame but not before driving each **SDI** low for at least one clock cycle. The **SDI** then becomes an input to the controller. Sometime during, but before the end of the Frame Sync at the end of the address frame, the codec starts driving the **SDI** low in preparation for normal operation. Normal link operation starts on the frame following the completion of the Address Frame. Codecs are required to actively drive a valid response field and to be ready to accept commands in this and subsequent frames.

Codecs with **SDI** signals that have not received an address assignment during the first 15 clock cycles of the Address Frame must behave as specified in Section 5.5.3.4.

5.5.3.3 Multi-SDI Codec Initialization

Codecs that use multiple **SDI** lines are required to follow the same initialization sequence as specified above for each connected **SDI** line. A multi-**SDI** codec must receive and store an address assignment for each of its **SDI** lines. When the codec is enumerated, software will determine which is the primary address, if appropriate to that codec, and which address(es) should be used for referencing verbs to this codec. During this codec enumeration, software also discovers and configures all other codec capabilities.

5.5.3.4 Un-initialized and Partially Initialized Codecs

Un-initialized Codecs

A codec that has not received any address assignment on any **SDI** may not participate in normal link operation but is required to ignore all **SDO** data, and drive its **SDI** signals low during normal link operation. Codecs are prohibited from re-trying an initialization request if they do not receive an address assignment during the Address Frame of codec initialization. The codec must wait for a subsequent link reset to further attempt initialization or connection to the Link.

Partially Initialized Codecs

A multi-**SDI** codec that has received an address assignment for some, but not all of its **SDI** signals, may or may not have limited operational capabilities. Partially initialized codecs must be capable of accepting commands and sending responses at the beginning of normal link operation. If **SDI** signal(s) required for this functionality have not been fully initialized, then this codec must behave as an un-initialized codec and ignore the link as specified above. If the codec is capable of accepting commands and sending responses, then it may provide the (limited) capabilities of which it is capable on the **SDI** signals that did complete enumeration. However, any codec functions that are dependent on **SDI** signals that did not complete initialization must be hidden from software as if the function does not exist; i.e., the related codec functional blocks must appear invisible to enumeration software. Furthermore, codecs are prohibited from re-trying an initialization request on **SDI**s that do not receive an address assignment during the Address Frame of codec initialization. The codec must wait for a subsequent link reset to further attempt initialization or connection to the Link.

5.6 Power Management

The High Definition Audio Architecture is designed to support all relevant power management features. In most cases, all power management state changes are driven by software, either through controller control registers or Command verbs to codecs. The exception to this is when a codec is put into a low power mode awaiting an external wake up event, such as a ring indication on a modem. In this case, the external wake event results in a power state change request on the Link as described below.

Whenever the Link is commanded to enter a low power state, it enters the link-reset state, as described in Section 5.5.1. This state is only exited in response to a software command and follows all link rules for exiting the link reset state.

Codecs, when put into a lower power state awaiting an external event, will post the occurrence of a wake event and request a power state change by signaling a power state change request and initialization request as described in Section 5.5.3.1. If **BCLK** and **SYNC** are running at the time of

the event, the codec will signal an unsolicited response as described in Section 5.5.3. If **BCLK** and **SYNC** are not running at the time, the codec will signal the power state change request and initialization request asynchronously by asserting **SDI** continuously until it detects the de-assertion of **RST#**, as shown in Figure 33.

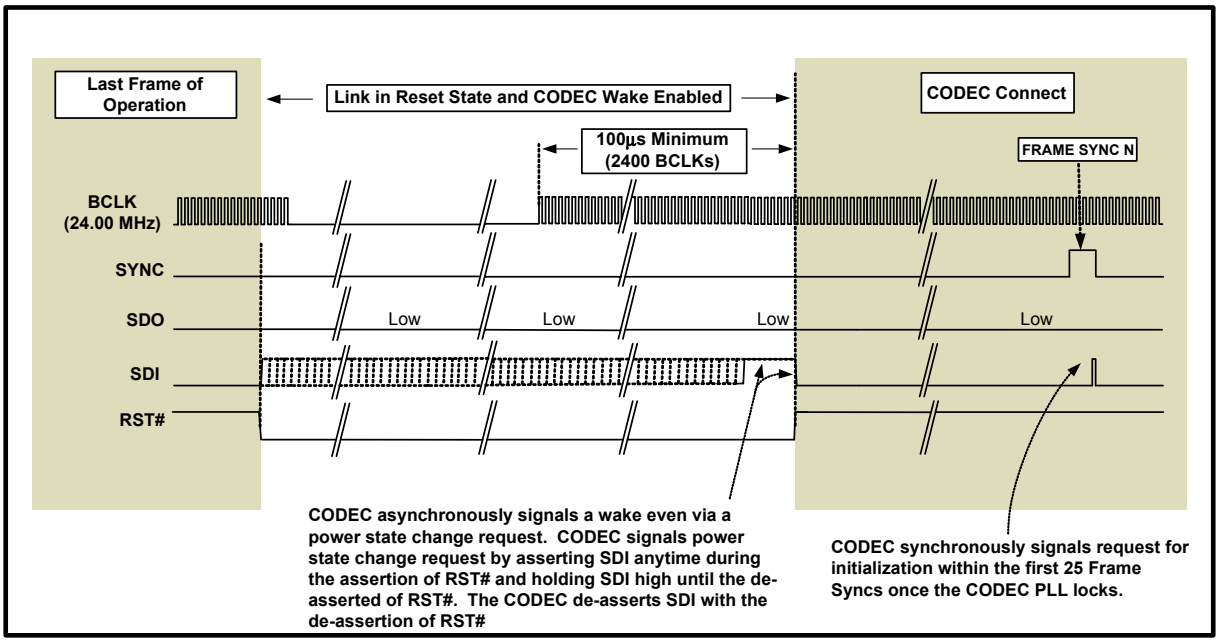

Figure 33. Resume From External Event

After signaling the power state change and initialization requests, the codec will be brought to a full power state by software via a command verb. See Section 4.5.9 for information regarding how software determines the source of the wake event when the Link resumes from a low power state.

6 Electrical Interface

6.1 Overview

This chapter defines the electrical characteristics and constraints of the High Definition Audio Link. It is divided into sections covering integrated circuit component and system specification and physical requirements. Each section contains the requirements that must be met by the respective product as well as the assumptions it may make about the environment provided. While every attempt was made to make these sections self-contained, there are invariably dependencies between sections so that it is necessary that vendors be familiar with all areas.

The electrical specification in this revision is defined for the 3.3V signaling environment.

6.1.1 3.3V to Low Voltage Transition

One goal of High Definition Audio is to provide a quick and easy transition from 3.3V to low voltage signaling on the electrical link.

The motherboard defines the signaling environment for the Link, whether it is 3.3V or the lower voltage. The 3.3V board is designed to work only with codec and controller components that are capable of 3.3V signaling. Similarly, the low voltage board is designed to work only with codec and controller components that are capable of low voltage signaling. However, a universal codec or controller may be designed such that it is capable of working in either of these two signaling environments. A later revision of this electrical specification would have the details for the low voltage signaling.

6.2 Component Specification

6.2.1 3.3V Signaling Environment

This section defines the electrical characteristics of components (codec and controller) for a 3.3V signaling scheme. The 3.3V components can be designed with standard CMOS I/O technology. Unless specifically stated otherwise, component parameters apply at the package pins not at bare silicon pads nor at card edge connectors.

6.2.1.1 DC Specifications

The DC specifications of the 3.3V codec and controller components are summarized in Table 58.

Table 58. 3.3V DC Specification

Symbol	Parameter	Condition	Min	Max	Units	Notes
Vcc	Supply Voltage		3.135	3.465	V	
Vih	Input High Voltage		0.65Vcc		V	
Vil	Input Low Voltage			0.35Vcc	V	
Voh	Output High Voltage	Iout =-500 µA	0.9Vcc		V	
Vol	Output Low Voltage	Iout = 1500 µA		0.10Vcc	V	
Iil	Input Leakage Current	0 < Vin < Vcc		±10	µA	1
Cin	Input Pin Capacitance			7.5	pF	
Lpin	Pin Inductance			20	nH	2

NOTES:

1. For **SDI** buffers (or in general any bi-directional buffer with tri-state* output), input leakage current also include hi-Z output leakage.
2. This is a recommendation, not an absolute requirement. The actual value should be provided with the component data sheet.

6.2.1.2 AC Specifications

The output driver on the Link must be able to deliver an initial voltage of at least Vil or Vih, respectively, at the receiver through the bus with known characteristic impedance and at the same time meeting signal quality requirements.

The minimum and maximum drive characteristics of output buffers are defined by the V/I curves. Figure 34 and Table 59 describe the **SDO** buffer AC drive specification whereas Figure 35 and Table 60 describe the AC drive specification of the SDI buffers. The AC drive specification for **SYNC**, **RST#**, and **BCLK** buffers is the same as that for **SDO**.

These curves should be interpreted as traditional DC transistor curves with the following exceptions: "DC drive point" is the only position on the curves at which steady state operation is intended, while the higher currents are only reached momentarily during bus switching transients. The "AC drive point" (real definition of buffer strength) defines the minimum instantaneous current required to switch the bus.

Adherence to these curves should be evaluated at worst case conditions. Minimum pull up curve is evaluated at minimum Vcc and high temperature. Minimum pull down curve is evaluated at minimum Vcc and high temperature. The maximum curve test points are evaluated at maximum Vcc and low temperature.

Inputs must be clamped to both ground and power rails. The clamp diode characteristics are also listed here for reference.

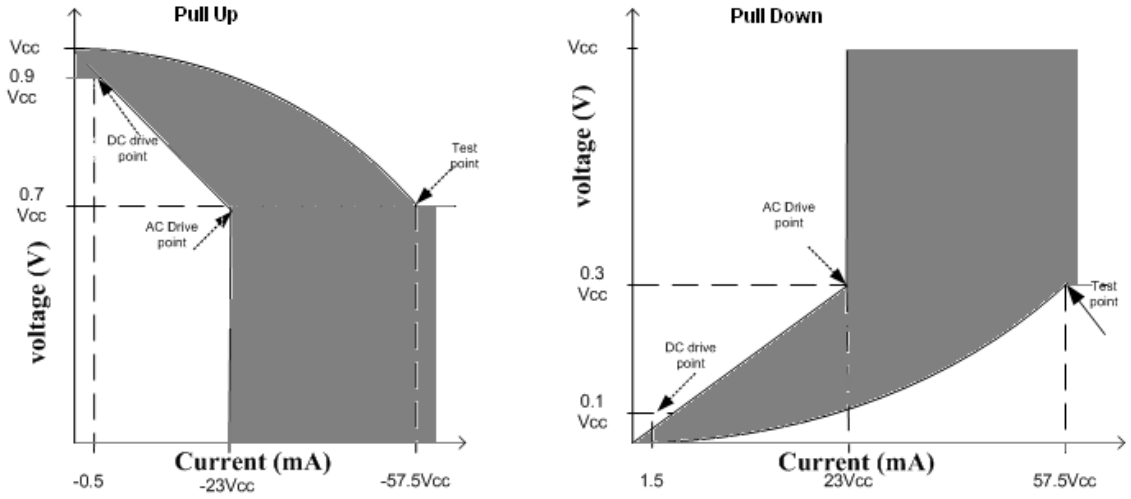

Figure 34. V/I Curves for SDO Buffers

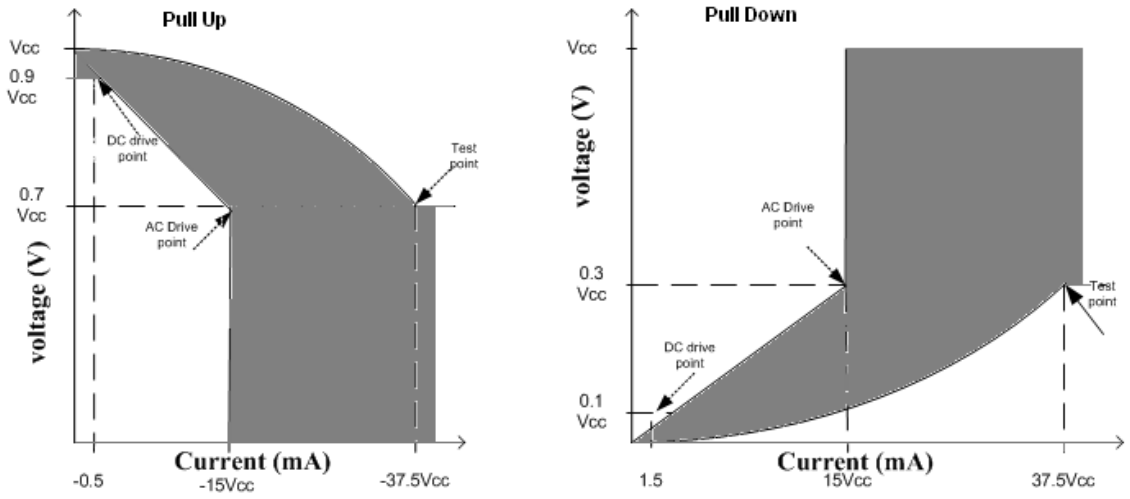

Figure 35. V/I Curves for SDI Buffers

Equation C

Ioh = (174.2/Vcc)*(Vout-Vcc)*(Vout + 0.4Vcc); for Vcc > Vout > 0.7 Vcc

Equation D

Iol = (273.8/Vcc)*Vout*(Vcc-Vout); for 0v < Vout < 0.3 Vcc

Equation E

Ioh = (113.6/Vcc)*(Vout-Vcc)*(Vout + 0.4Vcc); for Vcc > Vout > 0.7 Vcc

Equation F

Iol = (178.6/Vcc)*Vout*(Vcc-Vout); for 0v < Vout < 0.3 Vcc

Table 59. SDO Buffer AC Specification

Symbol	Parameter	Condition	Min	Max	Units
Ioh (AC)	Switching	0 < Vout < 0.7Vcc	-23Vcc		mA
	Current high	0.7Vcc < Vout < 0.9Vcc	-76.7(Vcc - Vout)		
		0.7Vcc < Vout < Vcc		Eqt'n C	
	(Test Point)	Vout = 0.7Vcc		-57.5Vcc	
Iol (AC)	Switching	Vcc > Vout >0.3Vcc	23Vcc		mA
	Current Low	0.3Vcc > Vout > 0.1Vcc	76.7Vout		mA
		0.3Vcc > Vout > 0		Eqt'n D	mA
	(Test Point)	Vout = 0.3Vcc		57.5Vcc	mA
Icl	Low Clamp	-3 < Vin < -1	-25 + (Vin + 1)/0.015		
	Current				mA
Ich	High Clamp	Vcc + 4 > Vin > Vcc + 1	25 + (Vin – Vcc - 1)/0.015		mA
	Current				
slew_r	Output rise	0.25Vcc to 0.75Vcc	1	3	V/ns
	Slew rate				(note1)
slew_f	Output rise	0.75Vcc to 0.25Vcc	1	3	V/ns
	Slew rate				(note1)

Table 60. SDI Buffer AC Specification

Symbol	Parameter	Condition	Min	Max	Units
Ioh (AC)	Switching	0 < Vout < 0.7vcc	-15Vcc		mA
	Current high	0.7Vcc < Vout < 0.9Vcc	-50(Vcc-Vout)		
		0.7Vcc < Vout < Vcc		Eqt'n E	
	(Test Point)	Vout = 0.7Vcc		-37.5Vcc	
Iol (AC)	Switching	Vcc > Vout > 0.3Vcc	15Vcc		mA
	Current Low	0.3Vcc > Vout > 0.1Vcc	50Vout		mA
		0.3Vcc > Vout > 0		Eqt'n F	mA
	(Test Point)	Vout = 0.3Vcc		37.5Vcc	mA
Icl	Low Clamp	-3 < Vin < -1	-25 + (Vin + 1)/0.015		
	Current				mA
Ich	High Clamp	Vcc+4 > Vin > Vcc+1	25 + (Vin – Vcc - 1)/0.015		mA
	Current				

Symbol	Parameter	Condition	Min	Max	Units
slew_r	Output rise	0.25Vcc to 0.75Vcc	1	3	V/ns
	Slew rate				(note 1)
slew_f	Output rise	0.75Vcc to 0.25Vcc	1	3	V/ns
	Slew rate				(note 1)

NOTES:

1. Slew rate is to be interpreted as the cumulative edge rate across the specified range (0.25Vcc to 0.75Vcc load for rise and 0.75Vcc to 0.25Vcc load for fall) rather than instantaneous rate at any point within the transition range. Section 6.2.1.4 specifies the load used to characterize the slew rates. This requirement for the slew rate applies to all the output buffers on the Link.

6.2.1.3 Maximum AC Ratings and Device Protection

All buffers on the Link should be capable of withstanding continuous exposure to the waveform shown in Figure 36. It is recommended that these waveforms be used as qualification criteria against which the long term reliability of each device is evaluated. Table 61 lists the parameters of the waveform. This level of robustness should be ensured by design; it is not intended that this waveform should be used as a production test.

These waveforms are applied with the equivalent of a zero-impedance voltage source, driving through a series resistor directly into each input or tri-stated output pin. The open-circuit voltage of the voltage source is shown in Figure 36 which is based on the worst case overshoot and undershoot expected in actual High Definition Audio buses. The resistor values are calculated to produce the worst case current into an effective internal clamp diode.

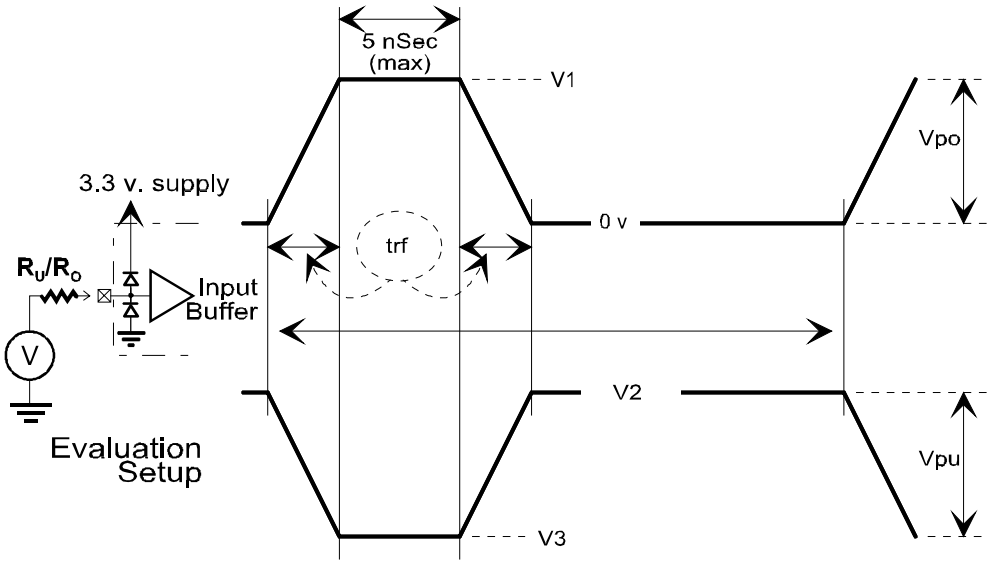

Figure 36. Maximum AC Waveforms for 3.3V Signaling

Note that in Figure 36, the test source resistor = 60 Ω.

Table 61. Parameters for Maximum AC Signaling Waveforms

Symbol	Parameter	Min	Max	Units
V1	Overshoot Voltage		6.3	V
V2	Undershoot initial Voltage		3.465	V
V3	Undershoot Voltage	-3		V
Vpu	Waveform peak-to-peak		6.465	V
Vpo	Waveform peak-to-peak		6.3	V
trf	Rise/fall time	1	3	V/ns
Freq	Frequency of AC rating waveform as applied to SDI input buffers		24	MHz
Freq	Frequency of AC rating waveform as applied to SDO input buffers		48	MHz

Note:

- The voltage waveform is supplied at the resistor shown in the evaluation setup not the package pin.
- Any internal clamping in the device being tested will greatly reduce the voltage levels seen at the package pin.

6.2.1.4 Measurements and Test Conditions

Figure 37 and Figure 38 define the timing parameters as measured at the controller and the codec respectively. The component test ensures that all timings are met with minimum clock slew rate (slowest edge) and minimum voltage swing. The design must ensure that minimum timings are also met with maximum clock slew rate (fastest edge) and maximum voltage swing. In addition, the design must ensure proper input operation for input voltage swings and slew rates that exceed the specified test conditions. The measurement conditions are summarized in Table 62. Figure 39 and Figure 40 specify the load used to characterize the slew rates and delay time. The reference point for all delay timing measurements is 0.5Vcc.

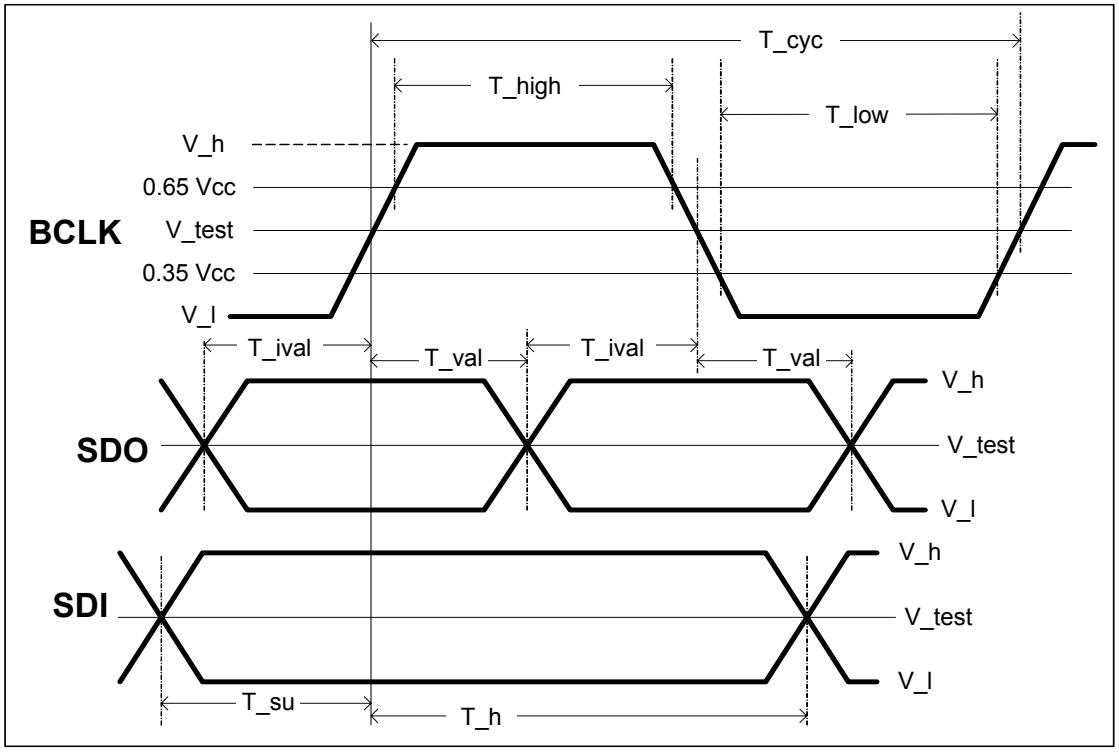

Figure 37. Timing Parameters as Measured at the Controller

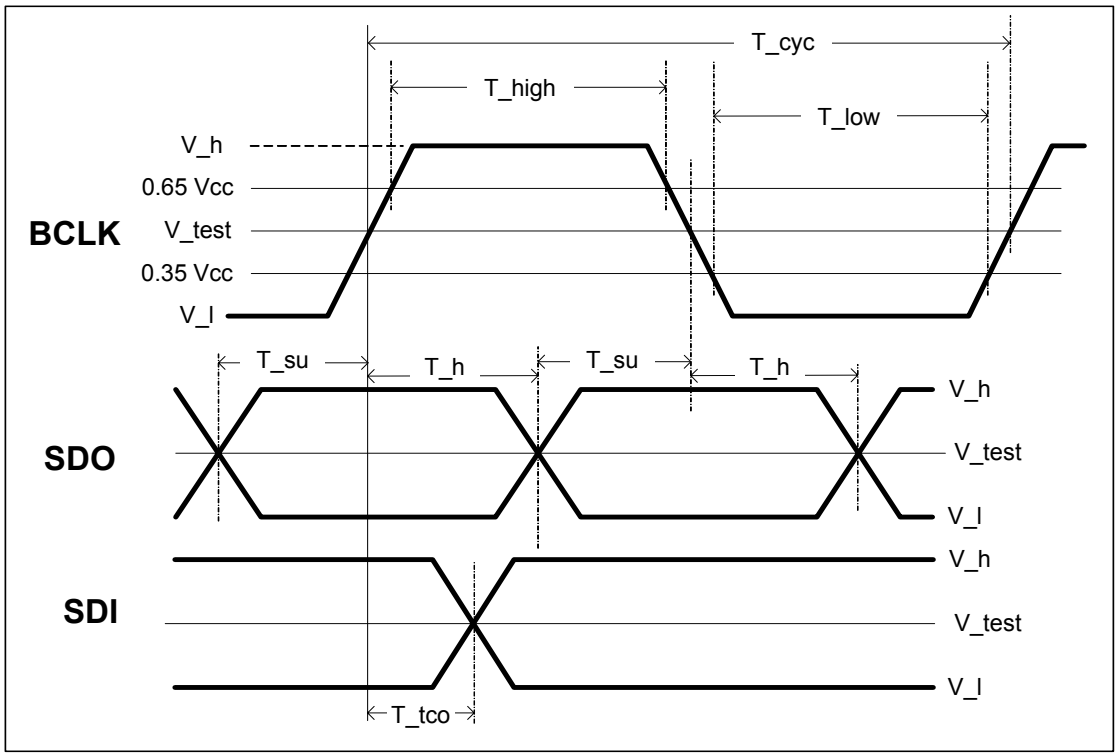

Figure 38. Timing Parameters as Measured at the Codec

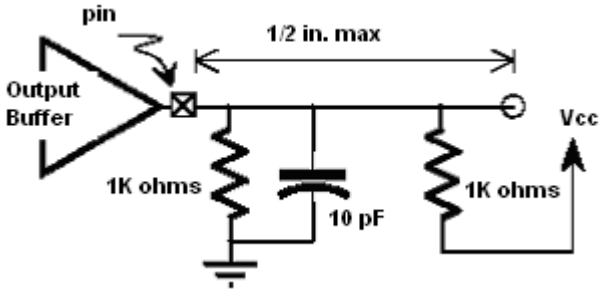

Figure 39. Slew Rate and Minimum Valid Delay

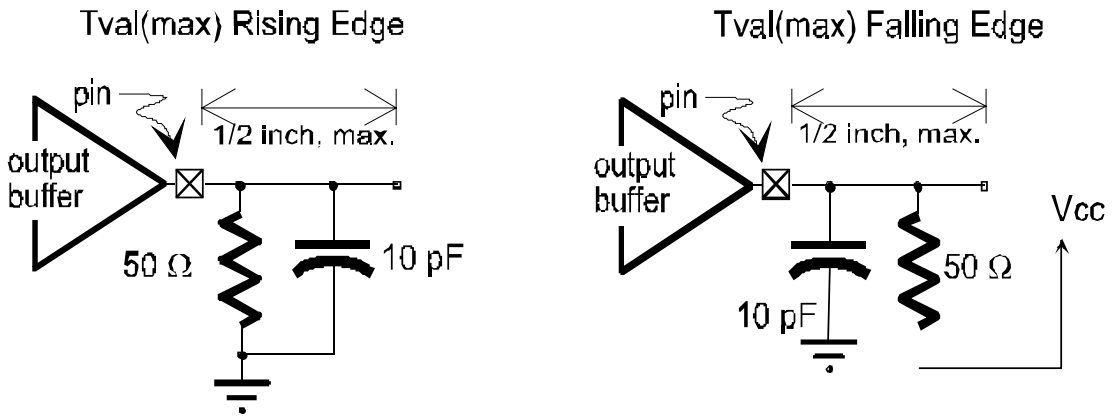

Figure 40. Maximum Valid Delay

Table 62. Measurement Condition Parameters

Symbol	3.3V Signaling	Units
V_h	0.75Vcc	V (note 1)
V_l	0.25Vcc	V (note 1)
V_test	0.5Vcc	V
V_max	0.5Vcc	V (note 2)

Notes:

1. Input test is done with 0.1Vcc of overdrive. V_h = Vih + 0.1 Vcc; V_l = Vil - 0.1Vcc.
2. V_max specifies the maximum peak to peak waveform allowed for measuring input timing.

6.2.2 Low Voltage Signaling Environment

The Low Voltage Signaling Environment will be defined in a subsequent revision of the High Definition Audio Specification.

6.2.3 Timing Specification

6.2.3.1 Timing Parameters

Table 63 describes the timing parameters at the controller interface and Table 64 describes the timing parameters at the codec interface. All the timing numbers are defined at the package pins of the corresponding interface. Rise and fall time, flight and delay time, setup and hold time listed here should be used together in the worst case scenario for modeling of the drivers on the Link. Setup and hold timing numbers for **SDO** are defined at every edge of the **BCLK** while those for **SDI** will be defined only at the rising edge of **BCLK**. This is due to the fact that **SDO** is double pumped while **SDI** is not. **SYNC** and **RST#** should be treated the same as **SDO** and hence have the same timing definitions as that of **SDO**. Section 6.2.1.4 describes the method and the loads used to characterize these timing parameters.

Table 63. Timing Parameters at the Controller

Symbol	Definition	Min	Typ	Max	Units	Notes
	BCLK frequency	23.9976	24.0	24.0024	MHz	3
T_cyc	Total period of BCLK	41.363	41.67	41.971	ns	3
T_high	High phase of BCLK	18.75		22.91	ns	1
T_low	Low phase of BCLK	18.75		22.91	ns	1
	BCLK jitter		150	300	ps	
T_ival	Time duration for which SDO is valid before the BCLK edge	7			ns	4
T_val	Time duration for which SDO is valid after the BCLK edge	7			ns	4
T_su	Setup for SDI at rising edge of the BCLK	15			ns	4
T_h	Hold for SDI at rising edge of the BCLK	0			ns	4

Table 64. Timing Parameters at the Codec

Symbol	Definition	Min	Typ	Max	Units	Notes
	Average BCLK frequency	23.9976	24.0	24.0024	MHz	3
T_cyc	Period of BCLK including jitter	41.163	41.67	42.171	ns	3
T_high	High phase of BCLK	17.5		24.16	ns	2
T_low	Low phase of BCLK	17.5		24.16	ns	2
	BCLK jitter		150	500	ps	
T_tco	Time after rising edge of the BCLK that SDI becomes valid	3		11	ns	4
T_su	Setup for SDO at both rising and falling edge of the BCLK	5			ns	4
T_h	Hold for SDO at both rising and falling edge of the BCLK	5			ns	4

NOTES:

1. 45/55 % is the worst case duty cycle at the controller as measured at v_test in Figure 37.
2. 42/58 % is the worst case duty cycle at the codec as measured at v_test in Figure 38.
3. This is the long term average frequency measured over 1 ms. Clock has a 100 ppm tolerance in the High Definition Audio Architecture.
4. The design should meet the timing requirements with the slew rate of the inputs in the range 1V/ns to 3 V/ns.

6.2.4 Vendor Provided Specification

The vendor of a system containing a High Definition Audio subsystem is responsible for electrical simulation of the High Definition Audio Link and components to ensure proper operation. To help facilitate this effort, component vendors are encouraged to make the following information available (It is recommended that component vendors make this information electronically available in the IBIS model format.):

- All parasitic parameters for the package and the die as seen at the codec package pin.
- Output static V/I curves and V/T under switching conditions. Two curves should be given for each output type used: one for driving high, the other for driving low. Both should show best-typical-worst curves.
- Input V/I characteristics. “Beyond-the-rail" response is critical, especially for inputs. The voltage range should span -3.3V to 6.6V for 3.3V signaling.
- Rise/fall slew rates for each output type.
- Complete absolute maximum data, including operating and non-operating temperature, DC maximums, etc.

6.3 System (Motherboard) Specification

System designers should be aware that with the increased driver strengths required to meet AC timings and signal integrity requirements, system routing may require system signal integrity improvement techniques, such as series resistors. It is the system designer’s responsibility to ensure compliance to this interface specification taking into account the fact that this is a multi-drop scheme. This can be accomplished by simulating and validating the applicable topology configurations and making sure signal quality and timing requirements are met.

This section covers the topologies that were investigated to validate this specification. While these topologies offer a proof-of-concept for a range of configurations, they are not intended to replace proper system validation of specific system designs.

6.3.1 Power Requirements

It is recommended that the motherboard connect all components on the Link with the same power supply voltages to the same power supply source and to a common ground plane to ensure minimum differences among their respective power supply and ground levels. Proper decoupling methods must be used to ensure stable power supply and ground. Separate supplies for I/O and logic will be specifically addressed in detail in the later revision of this specification which will include lower voltage signaling details.

6.3.2 System Timing Budget

The setup and hold numbers at the codec are dependent on the total skew introduced between **BCLK** and **SDO**. This should include signaling induced skews as well as skews due to trace mismatches on the board. Table 65 describes the limits for the total skew allowed on the system. Total skew allowed between **BCLK** and **SYNC** and **BCLK** and **RST#** is the same as that allowed for **BCLK** and **SDO** which is described below.

Table 65. Total Trace Mismatch

Description	Min	Max	Units	Notes
Total skew allowed between BCLK and SDO	-2	2	ns	1

Note:

1. This should be the total mismatch in the system including the motherboard and any applicable daughter cards or connectors.

Setup and hold numbers for **SDI** at the controller are dependent on the flight time from the controller to the receiver and vice-versa. Table 66 defines the maximum flight time allowed on the system.

Table 66. Maximum Trace Lengths

Description	Min	Max	Units	Notes
Flight time for BCLK from controller to the codec	0	7	ns	1
Flight time for SDI from the codec to the controller	0	7	ns	1

Note:

1. The maximum flight times in this table will dictate the total maximum allowed trace lengths between the codec and the controller.

6.3.3 Physical Requirements

6.3.3.1 System Board Impedance

This specification was validated using target trace impedance in the range of 55 Ω to 60 Ω with ± 15% allowed tolerance from target. A given system and any add-in cards for that system should pick a target tolerance in this range. The allowed tolerance is then applied to the chosen target impedance.

The system designer has two primary constraints in which to work:

- The length and signal velocity must allow a full round trip time on BCLK and SDI within the specified round trip propagation delay of 14 ns.
- The loaded impedance seen at any drive point on the network must be such that an Link output device (as specified by its V/I curve) can meet input device specifications. This includes loads presented by expansion boards.

6.3.3.2 Layout Guidelines

Topologies on desktop platforms have been analyzed with microstrip trace models while those on the mobile platforms have been simulated with both stripline and microstrip models. All the trace models used in the simulations to verify this specification were referenced to ground, although that is not a requirement. It is recommended that the reference plane be kept as consistent as possible. Changes in reference plane should have a bypass capacitor between these planes within 0.5 inches.

6.3.3.3 Trace Length Limits

Suggested trace length limits for each of the branches in the topologies are listed in Section 6.3.3.5 which describes the different desktop and mobile topologies.

6.3.3.4 Signal Loading

6.3.3.5 Topology Configurations

The High Definition Audio architecture uses multi-drop signaling scheme for **BCLK**, **SDO**, **SYNC**, and **RST#**. The list of system level topology configurations used to validate this specification has been defined here for reference. Figure 41 describes the topology configurations that have been simulated on a four layer microstrip desktop platform while Table 67 lists the possible configurations that can be derived from this. Figure 42 and Figure 43 describe the topology configurations that have been simulated on a typical multi-layer stripline and microstrip mobile platform. The number of codecs in the system will be limited by the number of SDI pins on the controller.

The intention is to illustrate possible multi-drop topologies. It is expected that system level simulations will be done using the actual trace and board models.

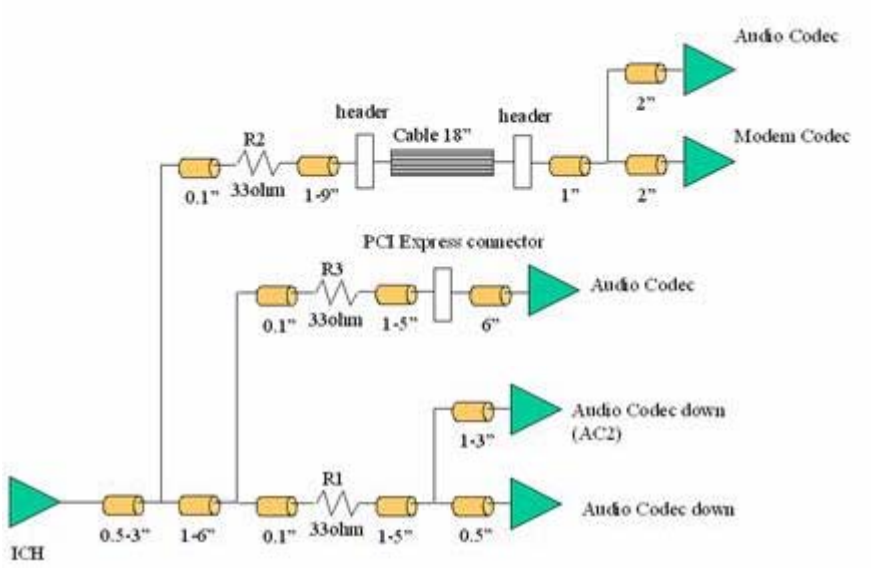

Figure 41. Desktop Platform Configurations

Table 67. System Simulation Setup for Desktop Configurations

Codec Configuration			R1	R2	R3	AC2	Notes
Down	After cable	After PCI					
1	2	1	Y	Y	Y		1 codec down on the motherboard, 2 after the cable on the back panel, and 1 on the PCI connector
2	2		Y	Y		Y	2 codecs down on the motherboard and 2 on the back panel or after the cable
2		1	Y		Y	Y	2 codecs on the motherboard and 1 on the PCI connector
1		1	Y		Y		1 codec on the motherboard and 1 on the PCI connector
1	2		Y	Y			1 codec on the motherboard and 2 on the back panel or after the cable
2			Y			Y	2 codecs down on the motherboard
1			Y				1 codec down on the motherboard

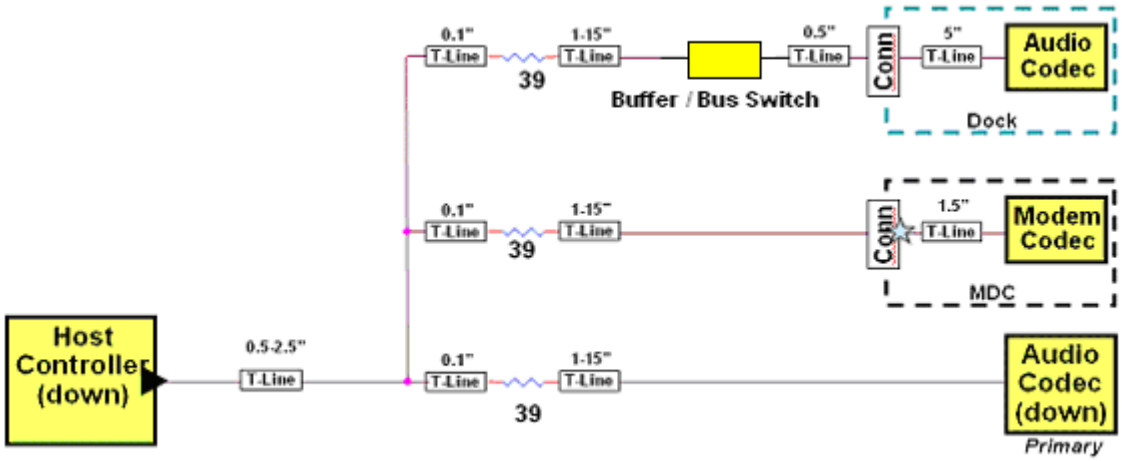

Figure 42. Mobile Platform Configurations

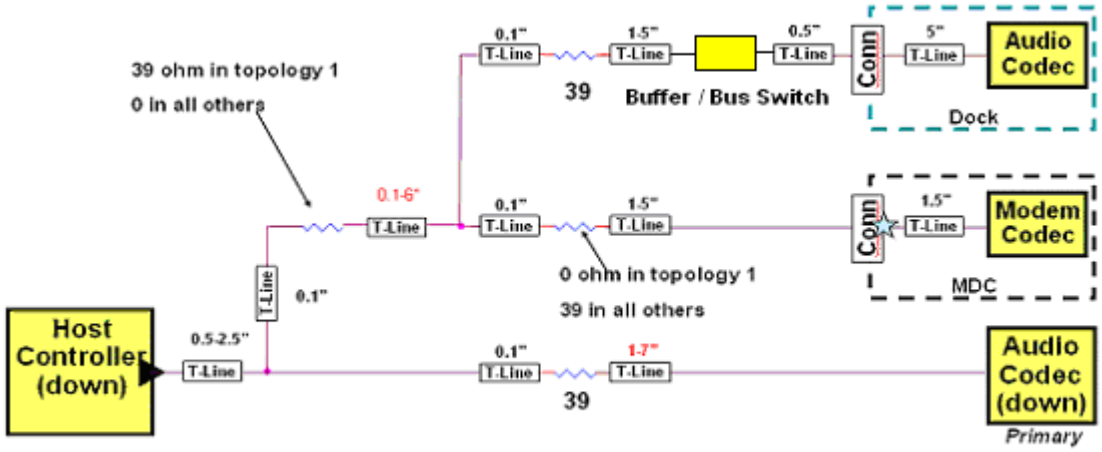

Figure 43. Branched Mobile Configuration

Note that the topology 1 in Figure 43 refers to a two codec configuration, one audio down and one codec on the MDC connector.

6.3.3.6 Hot Attach Mechanisms

An important feature for High Definition Audio is supporting hot attach. A typical example is connecting the Codec on the docking station to the Link while the Link is in operation. It is necessary that the part of the Link affected by the hot attach is isolated from the remainder of the Link during the connection. This is especially critical on the shared **BCLK**, **SDO**, **SYNC**, and **RST#** lines. If this isolation (during hot attach) is ignored, signal quality is highly degraded and would lead to functional failure of the other codecs on the Link.

There are two possible solutions to allow this isolation. The asynchronous solution, as shown in Figure 44, uses a high speed tri-state buffer as the isolation element during connect. The tri-state buffer is used on the **BCLK**, **SDO**, **SYNC**, and **RST#** signals. Since the buffer has a propagation delay, it consumes some of the 14-ns round trip delay allowed for **BCLK** and **SDI**. Consequently, this buffer should be carefully selected to be sure that the total delay meets the round trip flight time requirement. SDI is isolated by a simple FET bus switch since there is no shared signals here to be disrupted by the hot attach. The FET switch has a very small propagation delay. The switch prevents having an open line when the line is unconnected.

The synchronous solution to the hot attach problem is shown in Figure 45. This method requires more control logic, but the FET bus switches have much less propagation delay. This makes this scheme more suitable for topologies with longer flight times. The synchronous scheme ensures that there is no charge sharing during critical times that will cause glitches on the clock or invalid levels on control or data lines. Such glitches may result in functional failure of the Link.

Switching of the FET bus switch in this case is made synchronous to the **BCLK** transition; in this case, a positive edge triggered flip flops in the synchronous logic.

As in the case of BCLK shown in Figure 45, **SYNC**, **RST#**, and **SDO** also has bus switch in the branch that is connected to the hot attach. It is important that the delay through the flip flop to the enabling of the FET bus switch is less than half the clock period to ensure that the switch is enabled before the clock falls. This requires careful selection of the flip-flops and bus switches for speed. Since **DOCK** signal turn on is asynchronous to **BCLK**, a minimum of two flip-flops must be used in the enabling scheme; any additional flip-flop stages here would add to the turn on time. Clock charge-sharing problems are be avoided by using a weak pull-up resistor at the **DOCK** end of the Link. Values of R2 and C1 are selected such that the **DOCK** codec reset delay is 1 ms, given by the equation, R2C1ln [(Vcc - Vthl)/Vcc]. D1 diode used in the reset logic is needed for software initiated reset and should be selected such that the low to high threshold voltage is less than VCC – 700 mV.

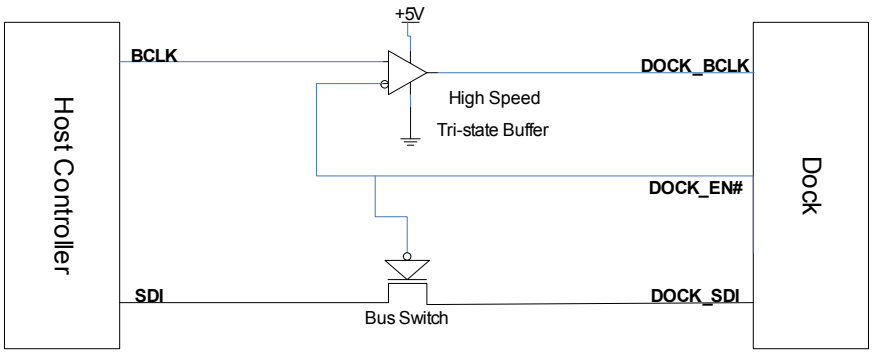

Figure 44. Hot Attach, Asynchronous Solution

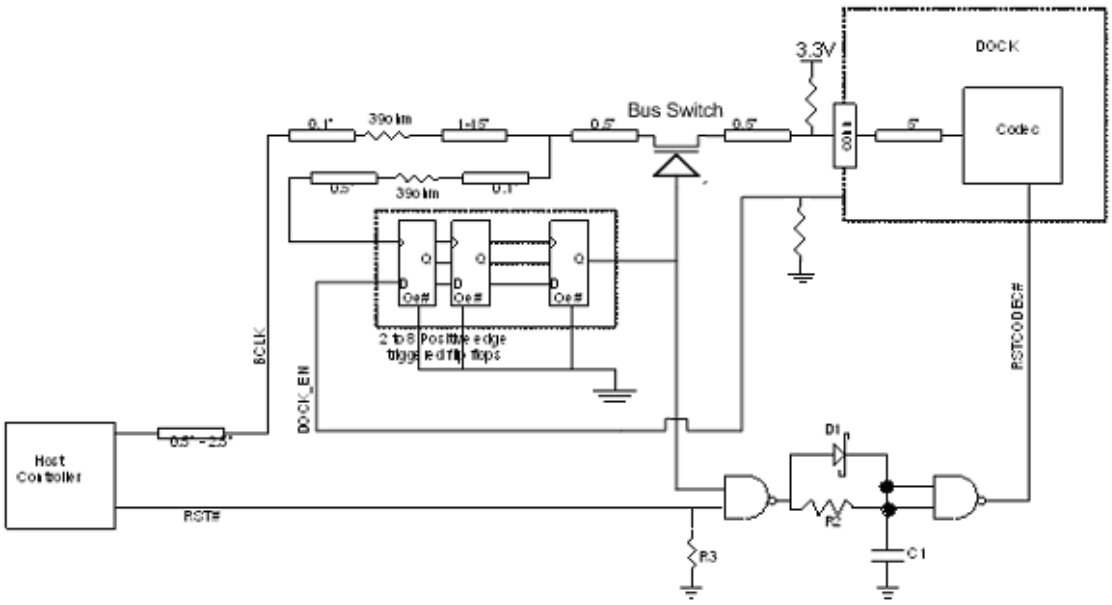

Figure 45. Hot Attach, Synchronous Solution

7 Codec Features and Requirements

This chapter defines the architectural and physical requirements for High Definition Audio codecs, including:

- Register definitions for codec parameters and controls
- Command and control Verb definitions
- Physical and mechanical (packaging) requirements
- Power Management requirements

Note that the term "codec" is often used herein to describe all devices (including modems) that connect to an High Definition Audio Controller via the High Definition Audio Link; the term is not limited strictly to audio codecs.

7.1 Codec Architecture

The High Definition Audio Specification defines a complete codec architecture that is fully discoverable and configurable so as to allow a software driver to control all typical operations of any codec. While this architectural objective is immediately intended for audio codecs, it is intended that such a standard software driver model not be precluded for modems and other codec types (e.g., HDMI, etc.). This goal of the architecture does not imply a limitation on product differentiation or innovative use of technology. It does not restrict the actual implementation of a given function but rather defines how that function is discovered and controlled by the software function driver.

7.1.1 Modular Architecture

The High Definition Audio Codec Architecture provides for the construction and description of various codec functions from a defined set of parameterized modules (or building blocks) and collections thereof. Each such module and each collection of modules becomes a uniquely addressable *node*, each parameterized with a set of read-only capabilities or *parameters*, and a set of read-write commands or *controls* through which that specific module is connected, configured, and operated.

The codec Architecture organizes these nodes in a hierarchical or tree structure starting with a single *root node* in each physical codec attached to the Link. The root node provides the "pointers" to discover the one or more *function group(s)* which comprise all codecs. A function group is a collection of directed purpose modules (each of which is itself an addressable node) all focused to a single application/purpose, and that is controlled by a single software function driver; for example, an audio function group (AFG) or a modem function group.

Each of these directed purpose modules within a function group is referred to as a *widget*, such as an I/O Pin Widget or a D/A Converter Widget. A single function group may contain multiple instances of certain widget types (such as multiple Pin Widgets), enabling the concurrent operation of several channels. Furthermore, each widget node contains a configuration parameter which identifies it as being "stereo" (two concurrent channels) or "mono" (single channel). Widgets within a single functional unit have a discoverable and configurable set of interconnection possibilities.

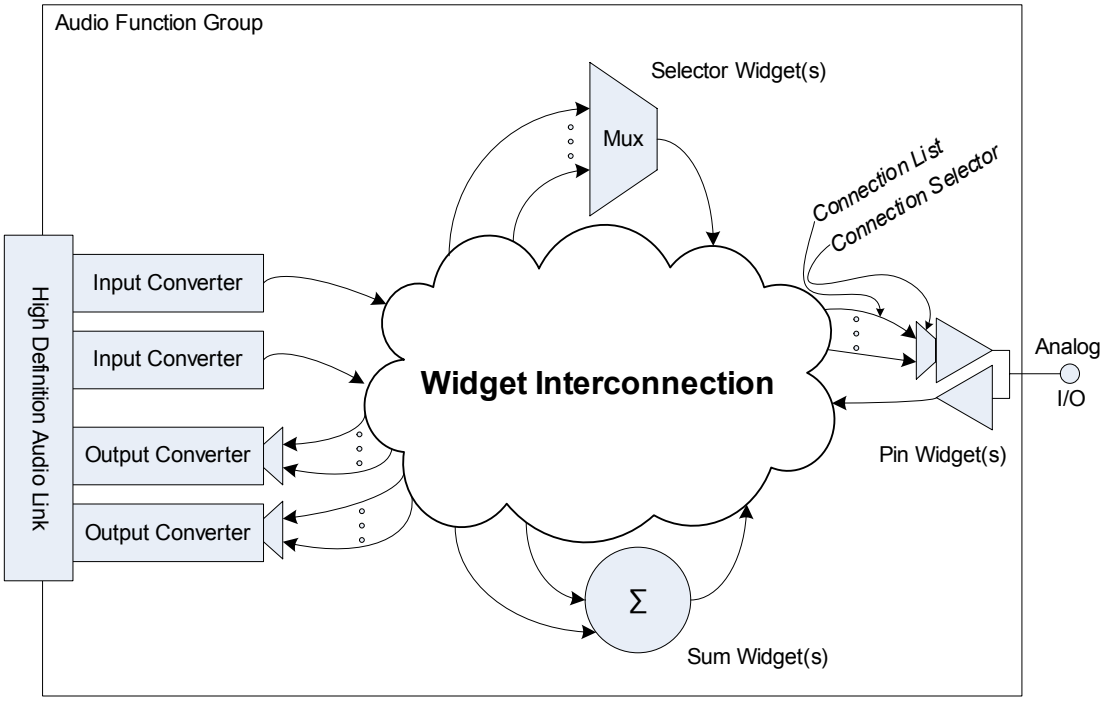

Figure 46. Module-Based Codec Architecture

Figure 46 illustrates an Audio Function Group, showing some of the defined widgets and the concept of their interconnection. Some of these widgets have a digital side that is connected to the High Definition Audio Link interface, in common with all other such widgets from all other function groups within this physical codec. Others of these widgets have a connection directly to the codec's I/O pins. The remaining interconnections between widgets occur on-chip, and within the scope of a single function group.

Each widget drives its output to various points within the function group as determined by design (shown as an interconnect cloud in Figure 46). Potential inputs to a widget are specified by a connection list (configuration register) for each widget and a connection selector (command register) which is set to define which of the possible inputs is selected for use at a given moment (see Section 7.1.2). The exact number of possible inputs to each widget is determined by design; some widgets may have only one fixed input while others may provide for input selection among several alternatives. Note that widgets that utilize only one input at a time (e.g., Pin Widget) have an implicit 1-of-n selector at their inputs if they are capable of being connected to more than one source, as shown in the Pin Widget example of Figure 46.

7.1.2 Node Addressing

Each node in the codec architecture (i.e., root node, function group nodes, and widget nodes) may be uniquely addressed to access its various parameters and controls for purposes of enumeration and run-time control. Each physical codec connected to the Link is assigned a unique *codec address* (CAd) at initialization (refer to Section 5.5.3.2), which is thereafter used as part of this addressing mechanism. Within a codec, nodes are organized in a three-level hierarchy: root node, function group nodes, and widget nodes. However, the addressing scheme for all of these nodes is

completely flat, and is defined to simplify the software discovery of the codec's capabilities (see Figure 47). Each node within a codec has a unique *node ID* (NID). The concatenation of CAd and NID provide a unique address allowing commands to reference a single specific node within the particular High Definition Audio subsystem.

The root node is the top level node of any codec and is always addressed as node zero (NID = 0) at the codec address (CAd) assigned to this codec. The root node contains device level information including the number of function groups in this codec and the NID of the first function group.

Each codec contains at least one function group. All function groups within the codec are identified with sequential NIDs; the root node specifies the beginning NID in this sequential series. Each function group contains several parameters, including the starting NID of its particular collection of widgets; all widgets associated with this function group must be numbered sequentially starting at the reported NID.

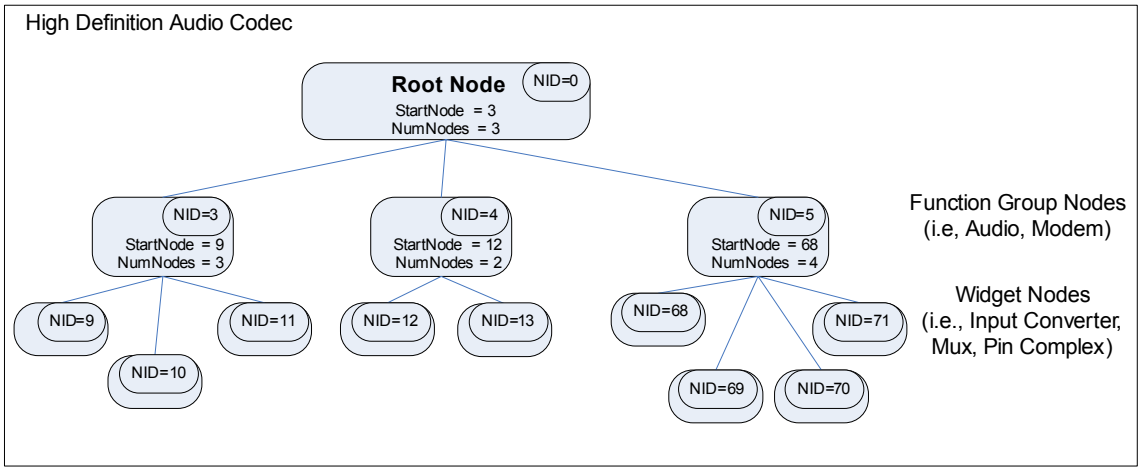

Figure 47. Codec Module Addressing Scheme

Nominally, a NID is represented in a "short-form" as a 7-bit integer. However, in very large codecs (e.g., containing more than 127 nodes), NIDs may be represented in a "long-form" as a 15-bit integer, as shown in Figure 48.

Each widget with inputs that are driven from the outputs of other widgets must have a Connection_List (parameter), or a list of NIDs that can be used as inputs. The number of entries (NIDs) in this list is specified in a Connect_List_Length register (parameter) as a 7-bit integer. The high order bit in the Connection List Length register indicates whether NIDs in this particular list will be represented in long-form or short-form (see Figure 48).

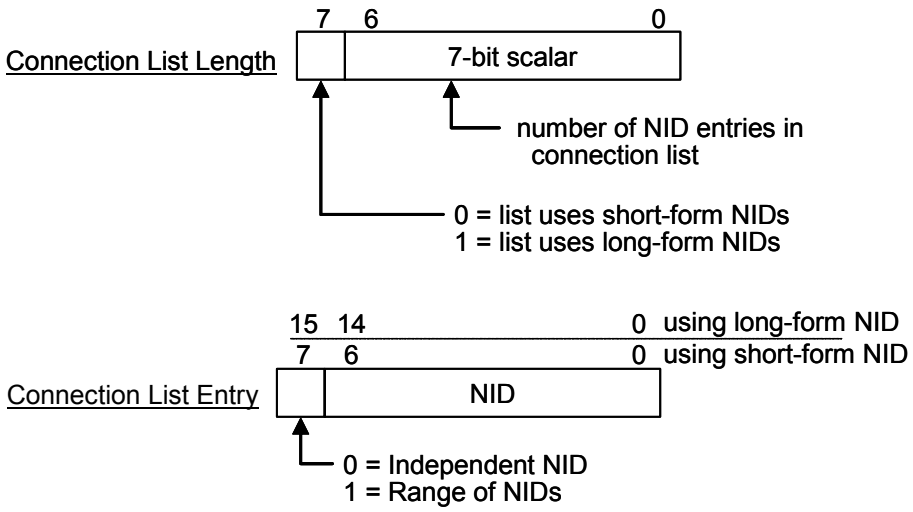

Figure 48. Connection Lists

Each entry in the connection list is one NID; it may be an independent NID, indicating a single node, or may be part of a 2-tuple of NIDs delineating a continuous range of nodes. The high order bit of each entry indicates how it is to be interpreted (see Figure 48). If the range indicator is set, that list entry forms a range with the previous list entry; i.e., if the range bit were set on the third list entry, then the second and third entries form a range, and the first entry is an independent NID. The range indicator may not be set in the first entry of a connection list nor may it be set on any two sequential list entries. The connection list thus provides a set of optional inputs to the node. A *connection selector* (control) allows run-time control over which input is used.

Identifying a range of nodes allows for a reduction of enumeration data space in some cases. For example, in specifying the possible connection of four input pins to a single A/D converter, the connectivity list may have four entries, each identifying one of the input pins, or, if the NIDs of the input pins were sequential, the connection list could be reduced in size to two entries specifying a range comprising the same four input pins.

Based on this addressing scheme, 12 of the 32 bits of a verb are used to address a specific node in the system, as shown in Figure 49. The high order four bits in any verb are the codec address and specify a physical codec on the Link. Bits 26-20 (7 bits) are the node ID. Bit 27 allows for an indirect addressing mechanism (to be specified) for codecs that have more than 127 nodes to address and, therefore, use the long form (15-bits) of node addressing. The low order 19 bits of a Verb contain the actual command and payload data.

A CAd of 15 (all ones) is reserved for broadcasting to all codecs. A codec must respond to this CAd = 15 as well as to the one it was assigned during the link initialization sequence. However, there are not currently any commands or verbs defined that are intended for use in a broadcast manner.

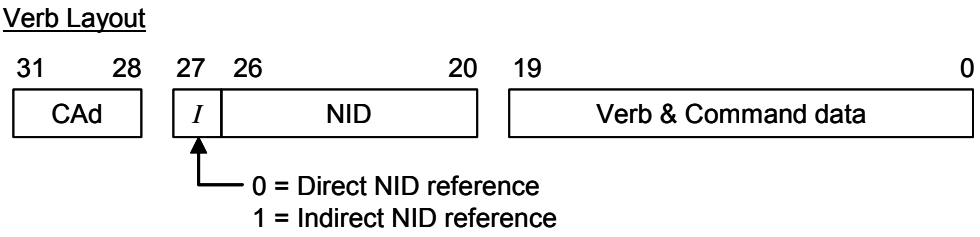

Figure 49. Verb Addressing Fields

7.1.3 Widget Interconnection Rules

All connected widgets must adhere to the following connection rules:

1. Widgets are exclusive to a single function group; they may never be shared between function groups.
2. Connection_List_Length = 1 means there is only one (hard-wired) input possible and, therefore, there is no Connection_Selector field. The actual connection is read from the Connection List as usual.
3. When a stereo widget is an input to another stereo widget, the L-channel of the source always drives the L-channel of the sink: R-channel to R-channel.
4. When a mono widget is an input to a stereo widget, the source always drives both L- and R-channels of the sink.
5. When a stereo widget is an input to a mono widget, only the L-channel of the source drives the sink. There is one exception to this rule, which allows a mono Mixer Widget to mix the L- and R-channels of a single stereo widget into a mono channel. This exception occurs only when the sink widget is:
   - A Mixer Widget, and
   - Is identified as "mono," and
   - Contains exactly one entry in its connection list, and
   - The single widget identified in its connection list (source) is "stereo."

   In this case, this mixer is required to have two inputs, the first being driven by the L-channel of the stereo source and the second by the R-channel.
6. Widgets may only interconnect within a function group. Widgets may not contain in their Connection Lists the addresses of widgets that are identified as being part of a different Function Group. Links between different function groups must be reported as external connections (Pin Complexes, in the example of an Audio Function Group) as if each function group was implemented in its own chip.

7.2 Qualitative Node Definition

This section provides a qualitative description of the standard modules or nodes that are defined in the High Definition Audio Codec Architecture and which may be used in various combinations to construct codec functions. Each of these modules or nodes is formally defined by its own set of parameters (capabilities) and controls (command and status registers): however, since some parameters and controls are formatted to be used with multiple nodes types, it is easier to first understand nodes at the qualitative level provided in this section. Thereafter, the exact data type,

layout, and semantics of each parameter and control is defined in Section 7.2.3.7. Finally, this is followed (Section 7.3.4.15) with tables specifying exactly which parameters and controls must be supported by which node types.

In the High Definition Audio Architecture, enumeration or discovery proceeds in a top down manner. After controller initialization, or upon a hot-plug event, the controller provides a list of the connected codecs each of which has been assigned a codec address (CAd). Within each codec, the root node (NID = 0) contains parameters specifying the function groups contained within the codec. Each function group, in turn, contains parameters specifying the widgets contained within the function group, and each widget contains parameters and controls used for its discovery, configuration, and operation.

The discovery and arrangement of widget types differs for each type of function group. It is the responsibility of the function driver to query the parameters of each of its widgets, to build a topological graph of widget connection options, and to configure and operate the overall function group.

7.2.1 Root Node

A codec's root node is always at NID = 0 and contains device level information serving as the starting point in the enumeration of the codec. The root node contains the following parameters:

- Vendor ID
- Device ID
- Revision ID
- Number of Function Groups within the codec including NID of the first one.

The root node has no accessible controls or commands.

7.2.2 Function Groups

A *function group* is a collection of widgets which are all common to a single application/purpose and which are controlled by a single "Function Driver." A codec contains one or more function group(s). While it is possible for a codec to contain more than one function group of a given type, this would not be typical. Currently defined function groups are:

- Audio Function Group
- Vendor Specific Modem Function Group
- HDMI Function Group (to be defined at a future time)

In addition to these function groups defined in this specification, it is possible for vendors to define other, proprietary function groups as part of a codec. Proprietary function group(s) will likely require proprietary parameters and controls and the verbs to access them. All of these (if used) must be defined within the High Definition Audio codec architectural model. Any vendor specific access method must strictly adhere to the defined meanings of verb address fields shown in Figure 49 ("CAd" and "NID" fields); vendor defined verbs may use only those verb encodings specified for such use (Section 7.3.5) together with the related payloads, all of which is contained in verb bits 19:0.

7.2.2.1 Audio Function Group

The Audio Function Group (AFG) contains the audio functions in the codec and is enumerated and controlled by the audio function driver. An AFG may be designed/configured to support an arbitrary number of concurrent audio channels, both input and output. An AFG is a collection of zero or more of each of the following types of widgets (note that all widget types or all combinations of nodes may not be supported by the standard software driver – consult your driver provider for details):

- Audio Output Converter
- Audio Input Converter
- Pin Complex
- Mixer (Summing Amp)
- 1-of-N Input Selector (multiplexer)

The parameters contained within an AFG node are generally:

- Subsystem ID
- Total number of widgets to be enumerated within the AFG, including NID of the first one
- Optional default converter and amplifier parameters that apply to all AFG widgets unless specifically over-ridden
- Number of GPIO pins on this codec that are assigned to the audio function
- Power management and other audio capabilities and parameters

The controls supported by an AFG node include reset and power management controls and read/write access to the GPIO pins.

7.2.2.2 Vendor Specific Modem Function Group

The Vendor Specific Modem Function Group (VSM-FG) contains the modem functions in the codec and is enumerated and controlled by a modem function driver. A VSM-FG is nominally designed/configured to support a single modem. It is required to provide certain standard parameters and controls sufficient for positive function group identification and a few basic controls including information to load the appropriate function driver. Beyond this, implementation of this function group is vendor defined.

The following specification-defined Verb encodings must be supported by all VSM-FG nodes:

- Get Parameter; encoding F00h, Section 7.3.3.1
- Get/Set Power State; encoding F05h/705h, Section 7.3.3.9
- Get/Set Unsolicited Response; encoding F08h/708h, Section 7.3.3.14
- Get SubSystem ID; encoding F20h, Section 7.3.3.30
- Function RESET; encoding 7FFh, Section 7.3.3.28

All remaining Verb encodings have no standard definition for the VSM-FG node and may be arbitrarily defined by the modem vendor. However, it is strongly recommended that the vendor attempt to use specification-defined Verbs where possible in order to avoid confusing duplications in Verb definitions.

In addition, the following specification-defined parameters must be supported by all VSM-FG nodes:

- Subordinate Node Count; parameter # 04h, Section 7.3.4.3
- Function Group Type; parameter # 05h, Section 7.3.4.4
- Supported Power States; parameter # 0Fh, Section 7.3.4.12

All remaining parameter numbers are reserved. If other parameters are desired, they must be accessed through a Verb other than "Get Parameters" (F00h).

Whether a VSM-FG implementation contains any vendor defined widgets, or is wholly implemented as a single node, is strictly a vendor choice. A vendor may also elect to define each of several 16-bit registers as "virtual widgets nodes" and thus utilize the node address space as register addresses within this particular function group. Alternately, the vendor may map various Verb encodings to registers, or devise other methods of accessing vendor specific control registers. Any vendor specific access method must strictly adhere to the defined meanings of Verb address fields shown in Figure 49 ("CAd" and "NID" fields); vendor defined Verbs may use only those verb encodings specified for such use (Section 7.3.5), together with the related payloads, all of which is contained in Verb bits 19:0.

7.2.3 Widgets

A *widget* is the smallest enumerable and addressable module within a function group. A single function group may contain several instances of certain widgets. For each widget, there is defined a set of standard parameters (capabilities) and controls (command and status registers). Again, each widget is formally defined by its own set of parameters (capabilities) and controls (command and status registers); however, since some parameters and controls are formatted to be used with multiple different widget types, it is easier to first understand widgets at the qualitative level provided in this section. Thereafter, the exact data type, layout, and semantics of each parameter and control are defined in Section 7.2.3.7. Currently defined widgets are:

- Audio Output Converter Widget
- Audio Input Converter Widget
- Pin Widget
- Mixer (Summing Amp) Widget
- Selector (Multiplexer) Widget
- Power Widget

In addition to these standard widgets defined in this specification, it is possible for vendors to define other proprietary widgets for use in any proprietary function groups they define.

7.2.3.1 Audio Output Converter Widget

The Audio Output Converter Widget is primarily a DAC for analog converters or a digital sample formatter (e.g., for S/PDIF) for digital converters. Its input is always connected to the High Definition Audio Link interface in the codec, and its output will be available in the connection list of other widget(s), such as a Pin Widget. This widget may contain an optional output amplifier, or a processing node, as defined by its parameters (Figure 50). Its parameters also provide information on the capabilities of the DAC and whether this is a mono or stereo (1- or 2-channel) converter. In order to save parameter space in an Audio Function Group incorporating several Audio Output Converter Widgets, the function group node may optionally contain defaults for many of the DAC and amplifier parameters; however, these defaults may be over-ridden with local parameters if necessary.

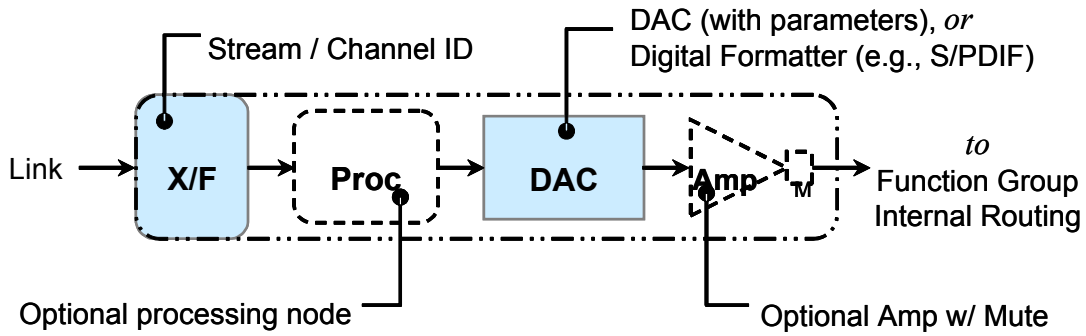

Figure 50. Audio Output Converter Widget

The Audio Output Converter Widget provides controls to access all its parametric configuration state, as well as to bind a stream and channel(s) on the Link to this converter. In the case of a 2-channel converter, only the "left" channel is specified; the "right" channel will automatically become the next larger channel number within the specified stream (see Section 7.3.3.11).

7.2.3.2 Audio Input Converter Widget

The Audio Input Converter Widget is composed primarily of an ADC for analog converters or a digital sample formatter (e.g., for S/PDIF) for digital converters. Its output is always connected to the Link interface in the codec, and its input will be selected from its own input connection list. This widget may contain an optional input amplifier, or a processing node, as defined by its parameters (Figure 51). Its parameters also provide information on the capabilities of the ADC, and whether this is a mono or stereo (1- or 2-channel) converter. In order to save parameter space in an Audio Function Group incorporating several Audio Input Converter Widgets, the function group node may optionally contain defaults for many of the ADC and amplifier parameters; however, these defaults may be over-ridden with local parameters if necessary.

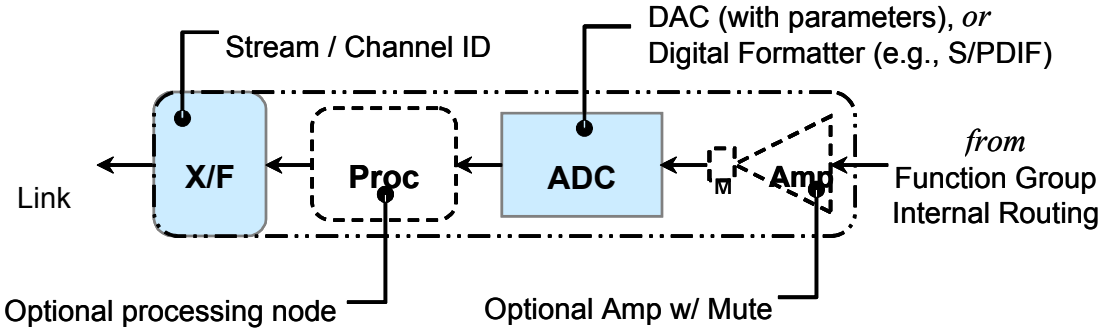

Figure 51. Audio In Converter Widget

The Audio Input Converter Widget provides controls to access all its parametric configuration state, as well as to bind a stream and channel(s) on the Link to this converter. In the case of a 2-channel

converter, only the "left" channel is specified; the "right" channel will automatically become the next larger channel number within the specified stream (see Section 7.3.3.11).

7.2.3.3 Pin Widget

The Pin Widget provides the external (analog or digital) connection for the audio and other function groups. A Pin Widget further includes those signals directly related to the external connections, such as jack sense and Vref control signals (Figure 52). However, GPIO pins are *not* identified as part of a Pin Widget but are a resource of the function group (see Sections 7.2.2.1 and 7.2.2.2). The Pin Widget's capabilities are highly parameterized defining optional support for:

- Input, output (or both), including the presence and capability of amplifier(s)
- Stereo or mono (1- or 2-channel)
- Plug (presence) detection
- Attached device impedance sensing
- VRef bias for microphone support

Every Pin Widget must contain a Configuration Default Register as defined in Section 7.3.3.31. In order to save parameter space in a function group incorporating several Pin Widgets, the function group node may optionally contain defaults for the amplifier parameters; however, these defaults may be over-ridden with local parameters if necessary.

The channel played on the external output pin (input to the Pin Widget) will be selected from its own input connection list; the channel on the external input pin will be available in the connection list of other widget(s), such as an Audio Input Converter Widget.

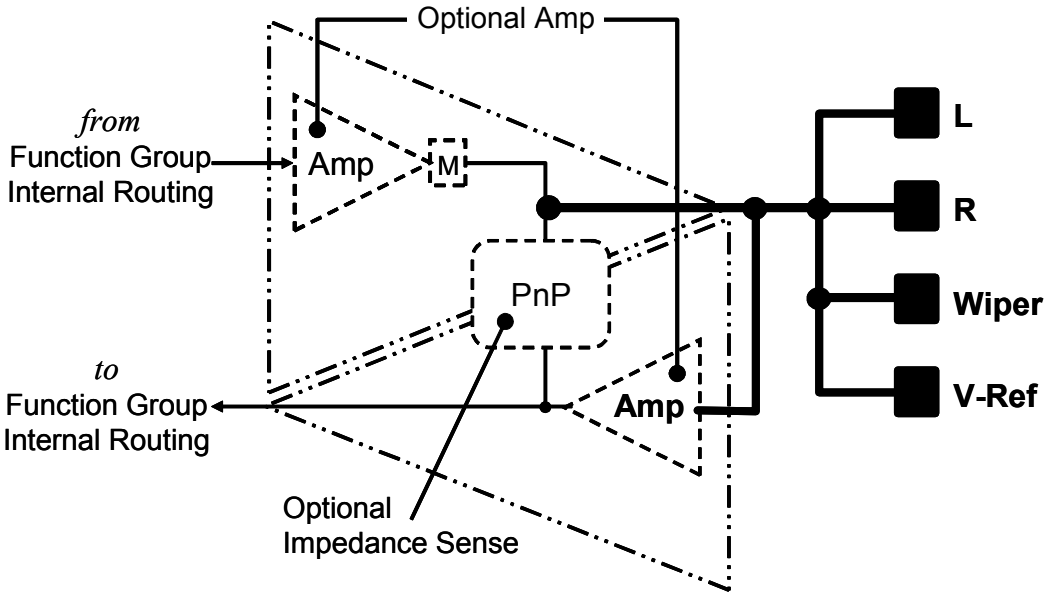

Figure 52. Pin Widget

7.2.3.4 Mixer (Summing Amp) Widget

The Mixer Widget provides the facility to arbitrarily mix multiple channels (sources). It has two or more inputs and one output. Each input has an optional input amplifier (including an optional mute), which is optional to all inputs collectively; i.e., they must all have (or not have) the amplifier/mute. The output also has an optional amplifier and optional mute. Input and output amplifiers may be separately defined, but all input amplifiers (when present) must have the same parameters. In the event that input amplifiers with differing parameters are needed, or if amplifiers are used on only some of the Mixer Widget inputs, then the Mixer Widget specifies no input amplifiers, and separate amplifiers (based on Selector Widgets) are created with appropriate connections to the Mixer Widget.

The Mixer Widget may be 1- or 2-channel (mono or stereo) with particular connections rules defined in Section 7.1.3. In order to save parameter space in an Audio Function Group incorporating several Mixer Widgets, the function group node may optionally contain defaults for the amplifier parameters which may be used to define the Mixer Widget amplifiers; however, these defaults may be over-ridden with local parameters if necessary.

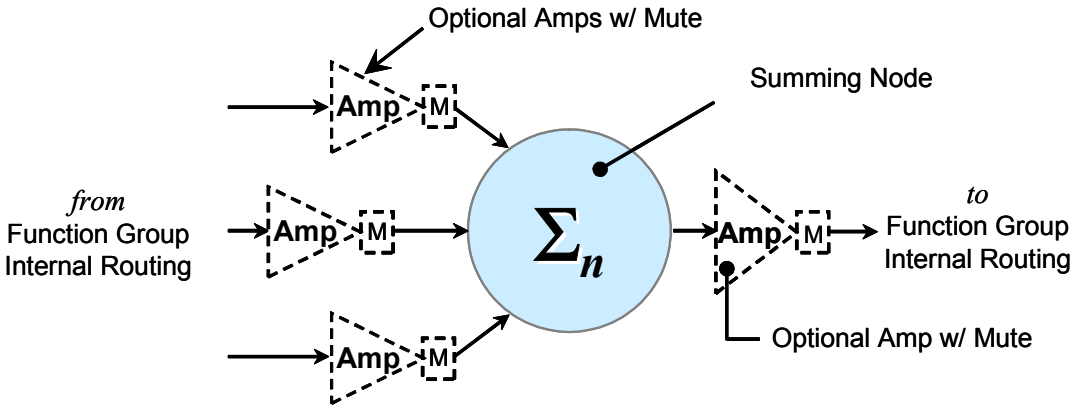

Figure 53. Mixer Widget

Inputs on the Mixer Widget are in its input Connection List and are hard wired (not selectable). Therefore, the Mixer Widget has no Connection Selector control. Input amplifiers/mutes are individually controlled by addressing their position (index) in the connection list. The Mixer Widget output will be available in the connection list of other widget(s).

7.2.3.5 Selector (Multiplexer) Widget

The Selector Widget provides a one-of-N signal selection. However, since the inputs to widgets generally have an implicit selector where needed, this widget may find infrequent use. A Selector Widget may be 1- or 2-channel and has one or more inputs and one output. Inputs on the Sum Widget are listed in its input Connection List; the output will be available in the Connection List of other widget(s). A selection among inputs is accomplished by a Connection Selector (control), the same as in other widgets with multiple inputs. The output may have an optional amplifier with mute. In order to save parameter space in an Audio Function Group incorporating several Selector Widgets, the function group node may optionally contain defaults for the amplifier parameters; however, these defaults may be over-ridden with local parameters if necessary.

Note that a degenerate (simple) Selector Widget (with only one input) allows a means of defining an arbitrarily placed amplifier or processing node.

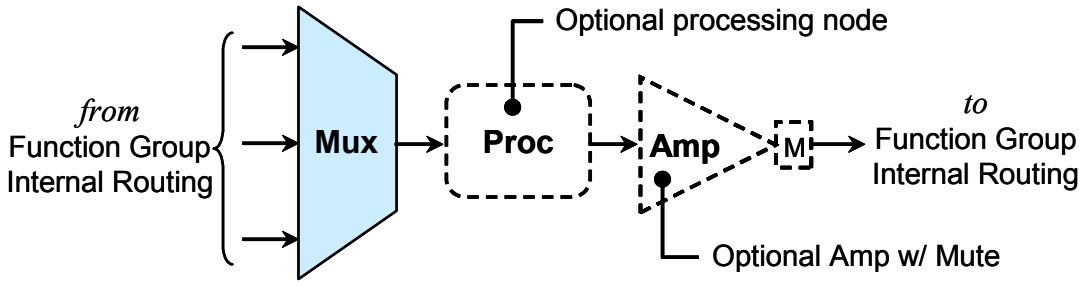

Figure 54. Selector Widget

7.2.3.6 Power Widget

The Power Widget provides a convenient means to optimize power management within the Audio Function Group by providing a single point of power state control for an arbitrary group of audio widgets. The Power Widget has no connections with other widgets but still uses its Connection List to specify the set of widgets associated with and controlled hereby. These associations are defined at design time and are not dynamic. Writing power state control to this widget will effectively place all associated widgets in the prescribed power state. However, in no circumstance may the Power Widget place any Audio Widget in a power state higher than the current power state of the Audio Function Group. Note that while the Power Widget does have a Connection List, it does not contain a Connection Selector, since there are no direct connections to other widgets.

7.2.3.7 Volume Knob Widget

The Volume Knob Widget provides for mechanical volume control of specified output pins. The physical nature of the volume control is not specified – it could be an analog wheel (pot.), pushbuttons, etc. – however, it does have a declared capability of being absolute (e.g., pot.) or relative (e.g., pushbuttons). This widget has a connection list that describes which other widgets (presumably Pin Widgets) have their volume controlled by the Volume Knob Widget; since no dynamic connections are formed, there is no Connection Selector.

The Volume Knob can be set by software to directly control the associated "slave" amplifiers, or to send an unsolicited response, allowing the function driver to "read" the Volume Knob and then adjust the associated "slave" amplifiers indirectly.

7.2.3.8 Beep Generator Widget

The tone or Beep Generator Widget is an option used to generate an approximated sine wave by dividing the 48-kHz frame marker by a programmable amount. When the beep generator is actively generating a tone, its output drives all Pin Widgets which are currently defined output pins in a method of the vendor's choice, either by switching the pin to the beep signal or by mixing the tone into the currently playing stream. This node is never listed on any other node's connection list. The actual vendor-defined connection only persists while the Beep Generator is actively generating a tone. This widget may contain an optional amplifier.

This Beep Generator feature is independent of any optional "PC Beep Pin" or "Analog Beep Pin" input which is intended to receive an externally generated tone or sound. The presence of such a beep input pin is not exposed to software, nor defined in this specification. If used, this type of beep input would be connected through the codec to output pins in a vendor defined way, but such a connection may be maintained *only* while the Link reset (**RST#**) is asserted.

7.3 Codec Parameters and Controls

The foregoing function groups and widgets are formally specified in terms of *parameters* and *controls,* all of which are accessed by *verbs*. Parameters return static read-only information about the capabilities or configuration options of the codec, function group, or widget. Parameters are accessed, either with the "Get_Parameter" verb or, for connection lists, the "Get_Connection_List_Entry" verb. Controls have an effect on the behavior of the codec, such as setting a converter's data format or causing a reset to happen. Most controls are readable and writable using separate verbs defined for accessing each specific control, but some controls (such as RESET) are essentially write-only, and there is no associated verb to read a value.

All verbs are sent on the Link in the Command Field (first 40 bits) of outbound frames (see Section 5.3.1). Figure 55 shows the format of the Command Field; the first 8 bits are reserved and are transmitted as 0's. Commands additionally contain a 4-bit codec address (CAd) which is assigned at initialization time and identifies the target codec, together with an 8-bit node ID (NID) that identifies the target node within the codec. The 20-bit verbs vary in format and are each documented in subsequent sections.

Bits 39:32	Bits 31:28	Bits 27:20	Bits 19:0
Reserved	CAd	NID	Verb

Figure 55. Command Field Format

There are two types of verbs: those with 4-bit identifiers and 16 bits of data payload and those with 12-bit identifiers and 8 bits of payload. Because of the limited encoding space for the 4-bit identifiers, they are used sparingly for operations which need to convey data to the codec in 16-bit payloads. The values 0x7 and 0xF are not legal values for 4-bit verbs, as they select the extended 12-bit identifiers.

The **SDO** signal contains exactly one verb envelope in each frame, as described in Section 5.3.1. Since there is no valid bit for verbs, as there is for responses, an invalid verb is defined as all 0's sent to NID = 0. That is, if verb bits [27:0] are all 0's, the verb is invalid; otherwise, it is a valid verb and must have an associated response.

7.3.1 Required Verb Response

Link protocol requires that all commands have a valid response. A codec must return a valid response on the frame following the frame on which a command addressed to it was received. Responses are sent on the Link in the Response Field (first 36 bits) of inbound frames (see Section 5.3.1).

Figure 56 shows the format of the Response Field; reserved bits are transmitted as 0's. A 1 in the Valid bit position indicates the Response Field contains a valid response, which the controller will place in the RIRB; a 0 indicates there is no response. A 1 in the UnSol bit position is meaningful only when the Valid bit is set and indicates that the response is Unsolicited rather than in reply to a verb. Unsolicited responses may be used to signal the occurrence of asynchronous codec events to the software driver. The 32 actual response bits vary in format and are each documented in subsequent sections.

Bit 35	Bit 34	Bits 33:32	Bits 31:0
Valid	UnSol	*Reserved*	Response

Figure 56. Response Field Format

For Get commands, the response delivers the requested control status; for most Set commands, the response is usually all 0's. In both cases, a response must always be sent in the subsequent frame; out of order or delayed responses are not allowed.

In cases where a Get command actually initiates a sequence that will not be finished by the next link frame, a response must still be sent but may require a "Ready" bit or indication to allow software to properly interpret the response and poll for a completed response later if necessary. The only currently defined case where this is required is the Pin Sense "Execute" verb (Section 7.3.3.15) which returns a valid response with a value of 0h immediately and must be queried at a later point when the Pin Sense operation is complete to determine the actual sensed value. In all other currently defined verbs, the valid response value is required on the following frame, and any vendor defined verbs must follow the same requirements.

Since verbs are function-specific, most verbs are not applicable to all nodes. If a verb is ever directed at a node for which that verb has no meaning (e.g., Set Stream/Channel directed at a Pin Widget), the node simply responds with all 0's. Hardware is not required to provide any form of error response to invalid or otherwise non-completing commands, other than returning a valid response of 0. In general, software may optionally verify completion of commands by reading back the command to see if the associated register holds the expected value.

7.3.2 Multiple SDI Operation

In certain applications, a codec may require more than a single **SDI** signal to support the required input bandwidth. In the simple case, there may be multiple functions on a single die, the aggregate input bandwidth of which exceeds the capacity of a single **SDI** signal. In this case, the designer may put multiple **SDI** signals on the codec but bind certain functions to a specific **SDI** signal. In this case, each group of functions *must* have its own root node; i.e., each **SDI** signal must be associated with a unique root node.

The more complex case comes when a single function's input bandwidth exceeds the capacity of a single **SDI** signal. In this case, multiple **SDI** signals must be fully sharable by any of the inputs within the function, and the function driver (software) alone binds input streams to specific **SDI** signals. In this case, there are multiple **SDI** signals associated with a single root node, and during codec initialization, each **SDI** is assigned a unique CAd (see Section 5.5.3.2). However, the hardware designer arbitrarily designates one of them as the "primary **SDI,**" and all verb responses must be returned on that single **SDI** signal. Verbs can (and will be) addressed to any of the CAds

associated with that root node, but the responses must always come back on the primary **SDI** signal. In this case, the function driver discovers which **SDI** signals are associated to a given root node, and therefore sharable, by requesting the "Device ID" on each of the valid codec addresses (CAd), and then observing which **SDI** signal the response is returned upon. After mapping the "primary **SDI**" or "primary CAd," as well as the associations between **SDI**s, it will carry out all further command and control functions on the primary address but will dynamically distribute the input data bandwidth between all associated **SDI**s. Unsolicited responses may be returned on any of the associated **SDI**s; however, it is recommended they also use only the primary **SDI**, unless unsolicited response latency is a factor.

In this case of multiple **SDI**s, the function driver assigns each input stream to a specific **SDI** as described in Section 7.3.3.12. However, this binding is dynamic and may change in mid-stream, as the function driver distributes the bandwidth of streams as they are created and terminated. It is therefore essential that the codec observe exactly the required timing of any assignment change. Note that these bindings are transparent to controllers that handle incoming traffic by stream ID and not by codec or **SDI** address.

7.3.3 Controls

7.3.3.1 Get Parameter

The **Get Parameter** command returns the value of the parameter indicated by the payload. Although all nodes support the Get Parameter verb, the specific parameters which may be requested depends on the node type. Specific requirement, by node type, on which parameters must be supported, is listed in Table 112, Section 7.3.4.15. The parameters are detailed in Section 7.3.4.

Command Options:

Table 68. Get Parameter Command

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F00h	The ID of the parameter to read	The parameter value

Applies to:

- All Nodes (Root Node, Function Group nodes, and Widget nodes)

7.3.3.2 Connection Select Control

For widgets that have multiple inputs, the **Connection Select** control determines which input is currently active. The index is in relation to the Connection List associated with the widget. The index is a zero-based offset into the Connection List.

If the Connection List Length value is 1, there is only one hard wired input possible and, therefore, there is no Connection Select control, and the verb for this control is not operable on that widget. The actual connection is read from the Connection List associated with the widget, as usual.

If an attempt is made to Set an index value greater than the number of list entries (index is equal to or greater than the Connection List Length property for the widget) the behavior is not predictable.

Command Options:

Table 69. Connection Select Control

	Verb ID	Payload (8 bits)	Response (32 bits)
Get	F01h	0	Bits 31:8 are 0 Bits 7:0 are the Connection index currently set
Set	701h	The Connection Index value to be set	0

Applies to:

- Input Converter
- Selector Widget
- Pin Complex
- Other

7.3.3.3 Get Connection List Entry

Returns the Connection List Entries at the index supplied. This command provides a way for the software to query the complete Connection List for a widget multiple entries at a time.

The requested index *n* is zero based. If the Long Form bit of the Connection List Length parameter (refer to Section 7.3.4.11) is 1, *n* must be even, and two long form Connection List entries will be returned. Therefore, requesting index 0 will return the values at offset 0 and 1, requesting index 2 will return the Connection List Entries at offset 2 and 3, etc. If the Long Form bit of the Connection List Length parameter is 0, *n* must be a multiple of four, and four short form Connection List entries will be returned. Therefore, requesting index 0 will return the values at offset 0, 1, 2, and 3, requesting index 4 will return the Connection List Entries at offset 4, 5, 6, and 7, etc.

If an index value is requested where the offset of the Connection List Entries would be greater than the number of Connection List Entries, the number of entries beyond the end of the list would be reported as 0's. For example, if the Connection List Length parameter is read as 0002h, indicating that there are two short form Connection List Entries in the list, requesting the Connection List Entry with *n* equal to 0 will return List Entry 0 in bits 7:0, List Entry 1 in bits 15:8, and 0's for the other list entries.

If an index value of greater than the number of list entries (*n* is equal to or greater than the Connection List Length property for the widget) the behavior is not predictable.

Command Options:

Table 70. Connection List Entry Control

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F02h	Offset index n	See response format below

Get Response With Short Form List Entries:

31:16	24:16	15:8	7:0
Connection List Entry n+3	Connection List Entry n+2	Connection List Entry n+1	Connection List Entry n

Figure 57. Response Format

Get Response With Long Form List Entries:

31:16	15:0
Connection List Entry n+1	Connection List Entry n

Figure 58. Response Format

The value returned contains the Connection List Entry *n* in the lower Word and Entry *n+1* in the upper Word. If the list contains an odd number of list entries and the highest legal index was requested, the top Word would contain 0, as there is no valid list entry at index *n+1*.

Applies to:

- Audio Input Converter
- Mixer Widget
- Selector Widget
- Pin Widget
- Power Widget
- Other Widget

7.3.3.4 Processing State

For widgets that implement processing capabilities in the widget, the **Processing State** command provides a way to control the behavior of the widget.

All widgets that implement processing must support the value 00 (Processing Off) and 01 (Processing On). The Processing Parameters can be queried to determine whether the Processing Benign state is supported.

Command Options:

Table 71. Processing State

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F03h	0	Bits 31:8 are 0 Processing State is in bits 7:0
Set	703h	Processing State	0

The **Processing State** field may have the following values:

00h: Processing Off: The widget must not make any modifications to the stream passing through it.

01h: Processing On: The widget may perform any processing it wishes.

02h: Processing Benign: The widget must not make any modifications to the stream passing through it which are not linear and time invariant. For instance, speaker compensation or EQ processing is acceptable, but 3-D spatialization processing is not. If the widget does not support a benign mode of processing, it treats the "Benign" setting the same as the "Off" setting.

03-7Fh: *Reserved*

80-FFh: Vendor Specific. A vendor may use these modes to control the processing of the widget.

Applies to:

- Input Converter
- Output Converter
- Selector Widget
- Pin Complex Widget
- Other Widget

7.3.3.5 Coefficient Index

For widgets that have loadable processing coefficients, the **Coefficient Index** is a zero-based index into the processing coefficient list which will be either read or written using the Processing Coefficient control.

When the coefficient has been loaded, the Coefficient Index will automatically increment by one so that the next Load Coefficient verb will load the coefficient into the next slot.

If Coefficient Index is set to be greater than the number of "slots" in the processing coefficient list, unpredictable behavior will result if an attempt is made to Get or Set the processing coefficient.

Command Options:

Table 72. Coefficient Index

	Verb ID	Payload (16 Bits)	Response (32 Bits)
Get	D	0	Bits 31:16 are 0 index n is in bits 15:0
Set	5	Index n	0

Applies to:

- Input Converter (that has loadable processing coefficients)
- Output Converter (that has loadable processing coefficients)
- Selector Widget (that has loadable processing coefficients)
- Pin Complex Widget (that has loadable processing coefficients)
- Other Widget (that has loadable processing coefficients)

7.3.3.6 Processing Coefficient

The Processing Coefficient control is available on widgets that have processing capabilities, and have loadable coefficients. If the "ProcWidget" bit in the Audio Widget Capabilities parameter is set, indicating that the node has processing capabilities, the Processing parameters can be queried to determine the number of coefficients that can be loaded.

Processing Coefficient loads the value *n* into the widget's coefficient array at the index determined by the Coefficient Index control. When the coefficient has been loaded, the Coefficient Index will automatically increment by one so that the next Load Coefficient verb will load the coefficient into the next slot.

Command Options:

Table 73. Processing Coefficient

	Verb ID	Payload (16 Bits)	Response (32 Bits)
Get	Ch	0	The coefficient value n (in the lower 16 bits)
Set	4h	The value n of the 16 bit coefficient to set	0

Applies to:

- Input Converter (that has loadable processing coefficients)
- Output Converter (that has loadable processing coefficients)
- Selector Widget (that has loadable processing coefficients)
- Pin Complex Widget (that has loadable processing coefficients)
- Other Widget (that has loadable processing coefficients)

7.3.3.7 Amplifier Gain/Mute

The **Amplifier Gain/Mute** control determines the gain (or attenuation) of one or several amplifiers in a widget. Depending on the combination of bits set, anywhere from a single channel on a single amplifier to both channels on both input and output amplifiers, may be programmed at the same time. Each amplifier setting must be read individually, however.

Command Options:

Table 74. Amplifier Gain/Mute

	Verb ID	Payload (16 Bits)	Response (32 Bits)
Get	Bh	See Note A	See Note B
Set	3h	See Note C	0

Note A: Get Payload Format:

15	14	13	12:4	3:0
Get Output/ Input	0	Get Left/ Right	0	Index

Figure 59. Amplifier Gain/Mute Get Payload

Get Output/Input controls whether the request is for the input amplifier or output amplifier on a widget. If 1, the output amplifier is being requested; if 0 the input amplifier is being requested.

Get Left/Right controls whether the request is for the left channel amplifier or right channel amplifier on a widget. If 1, the left amplifier is being requested; if 0, the right amplifier is being requested.

Index specifies the input index of the amplifier setting to return if the widget has multiple input amplifies, such as a Sum Widget. The index corresponds to the input's offset in the Connection List. If the widget does not have multiple input amplifiers, or if "Get Output/Input" is a 1 (requesting the output amplifier), these bits have no meaning and are ignored. If the specified index is out of range, all 0's are returned in the response.

Note B: Get Response Format:

31:8	7	6:0
0	Amplifier Mute	Amplifier Gain

Figure 60. Amplifier Gain/Mute Get Response

The Get response returns the Gain and Mute settings for the amplifier requested in the Get payload. If the "Get Payload," "Get Input/Output," "Get Left/Right," or "Index" bits are set to request the value of a Amplifier which does not exist in a widget, the response to the Get will be 00000000h.

Note C: Set Payload Format:

15	14	13	12	11:8	7	6:0
Set Output Amp	Set Input Amp	Set Left Amp	Set Right Amp	Index	Mute	Gain

Figure 61. Amplifier Gain/Mute Set Payload

Set Input Amp and **Set Output Amp** determine whether the value programmed refers to the input amplifier or the output amplifier in widgets which have both, such as Pin Widgets, Sum Widgets, and Selector Widgets. A 1 indicates that the relevant amplifier should accept the value being set. If both bits are set, both amplifiers are set; if neither bit is set, the command is effectively a no-op. Any attempt to set a non-existent amplifier is ignored.

Set Left and **Set Right** determine whether the left (channel 0) or right (channel 1) channel of the amplifier is being affected. A 1 indicates that the relevant amplifier should accept the value being set. If both bits are set, both amplifiers are set. Any attempt to set a non-existent amplifier is ignored. If the widget only supports a single channel, these bits are ignored and the value programmed applies to the left (channel 0) amplifier.

Index is only used when programming the input amplifiers on Selector Widgets and Sum Widgets, where each input may have an individual amplifier. The index corresponds to the input's offset in the Connection List. If the widget being programmed is not a Selector Widget or a Sum Widget, or the **Set Input Amp** bit is not set, this field is ignored. If the specified index is out of range, no action is taken.

Mute selects -∞ gain, but the hardware implementation will determine the actual degree of mute provided. A value of 1 indicates the mute is active. Generally, mute should default to 1 on codec reset, but there may be circumstances where mute defaults to its off or inactive state. In particular, if an analog PC Beep Pin is used, the mutes of associated outputs must default to 0 to enable the beep signal prior to the codec coming out of reset. This bit is ignored by any amplifier that does not have a mute option.

Gain is a 7-bit "step" value specifying the amplifier gain, the actual dB value of which is determined by the "StepSize," "Offset," and "NumSteps" fields of the Output Amplifier Capabilities parameter for a given amplifier. After codec reset, this "Gain" field must default to the "Offset" value, meaning that all amplifiers, by default, are configured to 0 dB gain. If a value outside the amplifier's range is set, the results are undetermined.

Applies to:

- Input Converter
- Output Converter
- Pin Complex
- Selector Widgets
- Mixer Widgets
- Other Widgets

7.3.3.8 Converter Format

The **Converter Format** control determines the format the converter will use. This must match the format programmed into the Stream Descriptor on the controller so that the data format being transmitted on the link matches what is expected by the consumer of the data.

All of the field definitions of the format structure defined in Section 3.7.1 are applicable to the audio converter. It is important to note that even though the converter widget is only stereo or mono, the CHAN field is still necessary to indicate to the converter the size of a block (see Figure 21) in cases where MULT is greater than 1.

Command Options:

Table 75. Converter Format

	Verb ID	Payload (16 Bits)	Response (32 Bits)
Get	Ah	0	Bits 31:16 are 0 format is in bits 15:0
Set	2h	Format	0

Bits for both the Set Payload and the Get Response are as defined in Section 3.7.1.

Applies to:

- Input Converter
- Output Converter

7.3.3.9 Digital Converter Control

The **Digital Converter Controls 1** and **2** operate together to provide a set of bits to control the various aspects of the digital portion of the Converter Widget. The S/PDIF IEC Control (SIC) bits are supported in one of two ways.

In the first case referred to as "Codec Formatted SPDIF," if a PCM bit stream of less than 32 bits is specified in the Converter Format control, then the S/PDIF Control bits, including the "V," "PRE," "/AUDIO," and other such bits are embedded in the stream by the codec using the values (SIC bits) from the Digital Converter Control 1 and 2. On an input PCM stream of less than 32 bits, the codec strips off these SIC bits before transferring the samples to the system and places them in the Digital Converter Control 1 and 2 for later software access.

In the second case referred to as "Software Formatted (or Raw) SPDIF," if a 32-bit stream is specified in the Converter Format control, the S/PDIF IEC Control (SIC) bits are assumed to be embedded in the stream by software, and the raw 32-bit stream is transferred on the link with no modification by the codec. Similarly, on a 32-bit input stream, the entire stream is transferred into the system without the codec stripping any bits. However, the codec must properly interpret the Sync Preamble bits of the stream and then send the appropriately coded preamble. The IEC60958 specification, Section 4.3, "Preambles," defines the preambles and the coding to be used. Software will specify the "B," "M," or "W" (also known as "X," "Y," or "Z") preambles by encoding the last four bits of the preamble into the Sync Preamble section (bits 0-3) of the frame. The codec must examine the bits specified and encode the proper preamble based on the previous state. The previous state is to be maintained by the codec hardware. For more information on Preamble Coding, consult Section 4.3 of the IEC 60958 specification.

Table 76. SPDIF Sync Preamble Bits

Preamble Bits Set by Software (Bits 3:0 of Frame)	Preamble Coding	
	Previous State = 0	Previous State = 1
1000b (“B” or “Z”)	11101000	00010111
0010b (“M” or “X”)	11100010	00011101
0100b (“W” or “Y”)	11100100	00011011

Command Options:

Table 77. S/PDIF Converter Control 1 and 2

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F0Dh[5]	0	Bits 31:16 are 0 Bits 15:0 are SIC bits
Set 1	70Dh	SIC bits [7:0]	0
Set 2	70Eh	SIC bits [15:8]	0

15	14:8	7	6	5	4	3	2	1	0
Rsvd	CC[6:0]	L	PRO	/AUDIO	COPY	PRE	VCFG	V	DigEn

Figure 62. S/PDIF IEC Control (SIC) Bits

CC[6:0] (Category Code): Programmed according to IEC standards, or as appropriate.

L (Generation Level): Programmed according to IEC standards, or as appropriate.

PRO (Professional): 1 indicates Professional use of channel status; 0 indicates Consumer.

/AUDIO (Non-Audio): 1 indicates data is non-PCM format; 0 indicates data is PCM.

COPY (Copyright): 1 indicates copyright is asserted; 0 indicates copyright is not asserted.

PRE (Preemphasis): 1 indicates filter preemphasis is 50/15 μs; 0 preemphasis is none.

VCFG (Validity Config.): Determines S/PDIF transmitter behavior when data is not being transmitted. When asserted, this bit forces the de-assertion of the S/PDIF “Validity” flag, which is bit 28 transmitted in each S/PDIF subframe. This bit is only defined for Output Converters and is defined as Reserved, with a Read Only value of 0 for Input Converters.

- If “V” = 0 and “VCFG”=0, then for each S/PDIF subframe (Left and Right) bit[28] “Validity” flag reflects whether or not an internal codec error has occurred (specifically whether the S/PDIF interface received and transmitted a valid sample from the High Definition Audio Link). If a valid sample (Left or Right) was received and successfully transmitted, the “Validity” flag should be 0 for that subframe. Otherwise, the “Validity” flag for that subframe should be transmitted as “1.”

[5] The Verb Code F0Eh is reserved for S/PDIF Converter Control 2 and may never be reassigned to anything else. However, it need not be implemented since standard software drivers must never use it. If a codec elects respond to this code, the response must be identical in all respects to the response to Verb Code F0Dh.

- If "V" = 0 and "VCFG" = 1, then for each S/PDIF subframe (Left and Right), bit[28] "Validity" flag reflects whether or not an internal codec transmission error has occurred. Specifically, an internal codec error should result in the "Validity" flag being set to 1. In the case where the S/PDIF transmitter is not receiving a sample or does not receive a valid sample from the High Definition Audio Controller (Left or Right), the S/PDIF transmitter should set the S/PDIF "Validity" flag to 0 and pad each of the S/PDIF "Audio Sample Word" in question with 0's for the subframe in question. If a valid sample (Left or Right) was received and successfully transmitted, the "Validity" flag should be 0 for that subframe.
- If "V" = 1 and "VCFG" = 0, then each S/PDIF subframe (Left and Right) should have bit[28] "Validity" flag = 1. This tags all S/PDIF subframes as invalid.
- "V" = 1 and "VCFG" = 1 state is reserved for future use.
- Default state, coming out of reset, for "V" and "VCFG" should be 0 and 0 respectively.

V (Validity): This bit affects the "Validity flag," bit[28] transmitted in each subframe, and enables the S/PDIF transmitter to maintain connection during error or mute conditions. The behavior of the S/PDIF transmitter with respect to this bit depends on the value of the "VCFG" bit.

DigEn (Digital Enable): Enables or disables digital transmission through this node. A 1 indicates that the digital data can pass through the node. A 0 indicates that the digital data is blocked from passing through the node, regardless of the state.

Applies to:

- Input Converter
- Output Converter

7.3.3.10 Power State

The **Power State** control determines the power state of the node to which it refers. There is no required power saving or maximum allowed power in any of the low power states; rather these states allow the vendor to reduce power by as much or as little as desired to meet their customer needs. However, power must never be reduced to a given circuit in a manner that would be inconsistent with the specified power recovery requirements of that power state.

Command Options:

Table 78. Power State

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F05h	0	Bits 31:8 are 0 PS-Act is in bits 7:4 PS-Set is in bits 3:0
Set	705h	PS-Set in bits 0:3 bits 4:7 are 0	0

PS-Set is a PowerState field which defines the current power setting of the referenced node. If the referenced node is of any type other than a Function Group node, the actual power state is a function of both this setting and the PowerState setting of the Function Group node under which the currently referenced node was enumerated (is controlled).

PS-Act is a PowerState field which indicates the actual power state of the referenced node. Within a Function Group type node, this field will always be equal to the PS-Set field (modulo the time required to execute a power state transition). Within any other type of node, this field will be the

lower power consuming state of either a) the PS-Set field of the currently referenced node or b) the PS-Set field of the Function Group node under which the currently referenced node was enumerated (is controlled).

All PowerState fields are defined as follows:

PowerState[1:0]:

00: Node Power state (D0) is fully on.
01: Node Power state (D1) allows for (does not require) the lowest possible power consuming state from which it can return to the "fully on" state (D0) within 10 ms, excepting analog pass through circuits (e.g., CD analog playback) which must remain fully on.
10: Node Power state (D2) allows for (does not require) the lowest possible power consuming state from which it can return to the "fully on" state (D0) within 10 ms. For modems, this is the "wake on ring" power state.
11: Node Power state (D3) allows for (does not require) lowest possible power consuming state under software control. Note that any low power state set by software must retain sufficient operational capability to properly respond to a subsequent software Power State command.

PowerState[3:2]: *Reserved,* always 0.

While Function Group nodes (Audio Function, Modem Function, etc.) and Power Widget nodes must support this control, other widget nodes may optionally implement this control to provide more fine-grained power management of the codec. For Audio Widgets, such as Input Converter or Output Converter Widgets, the Audio Widget Capabilities parameter (see Section 7.3.4.6) will define whether this control is supported.

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group
- Power Widget
- Input Converter (Optional)
- Output Converter (Optional)
- Selector Widget (Optional)
- Mixer Widget (Optional)
- Pin Complex (Optional)

7.3.3.11 Converter Stream, Channel

Converter Stream, Channel controls the link Stream and Channel with which the converter is associated.

Command Options:

Table 79. Converter Control

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F06h	0	Bits 31:8 are 0 Stream is in bits 7:4 Channel is in bits 3:0
Set	706h	Stream is in bits 7:4 Channel is in bits 3:0	0

Stream is an integer representing the link stream used by the converter for data input or output. 0000b is stream 0, 0001b is stream 1, etc. Although the link is capable of transmitting any stream number, by convention stream 0 is reserved as unused so that converters whose stream numbers have been reset to 0 do not unintentionally decode data not intended for them.

Channel is an integer representing the lowest channel used by the converter. If the converter is a stereo converter, the converter will use the channel provided, as well as channel+1, for its data input or output.

Applies to:

- Input Converter
- Output Converter

7.3.3.12 Input Converter SDI Select

Input Converter SDI Select controls which High Definition Audio Link **SDI** signal is used to transmit all data (samples) of a given stream; this control is only relevant in codecs which support multiple SDI signals (refer to Section 7.3.2). The control is addressed to, and set in, any Input Converter to which channel zero of a stream has been assigned. The codec then transmits *all channels* of the associated stream on the designated **SDI** signal, since a single stream may never be broken across multiple **SDI**s. This implies that Input Converters not assigned to channel zero of a stream must slave this control off that of the Input Converter that is assigned to channel zero of the same stream. This control is not valid if it is referenced to an Input Converter not assigned to channel zero of a stream; in this case, 0's are returned and no action is taken.

Command Options:

Table 80. SDI Select

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F04h	0	Bits 31:4 are 0 SDI-Select is in bits 3:0
Set	704h	bits 7:4 are 0 SDI-Select is in bits 3:0	0

SDI-Select is an integer representing the hardware assigned CAd (refer to Section 5.5.3.2) of the **SDI** signal to be used by the converter for data input of its assigned stream.

Note that SDI assignment may be dynamically changed; operation of the Link frame in which this command is received will remain unchanged, but any **SDI** (re)assignment will be effective in the immediately subsequent frame – the one in which the command response would be returned.

7.3.3.13 Pin Widget Control

Pin Widget Control controls several aspects of the Pin Widget.

Command Options:

Table 81. Enable VRef

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F07h	0	Bits 31:8 are 0 Bits 7:0 are PinCntl
Set	707h	Bits 7:0 are PinCntl	0

PinCntl format:

7	6	5	4:3	2:0
H-Phn Enable	Out Enable	In Enable	Rsvd	VRefEn

Figure 63. PinCntl Format

H-Phn Enable disables/enables a low impedance amplifier associated with the output. The value 1 enables the amp. Enabling a non-existent amp is ignored.

Out Enable allows the output path of the Pin Widget to be shut off. The value 1 enables the path. Enabling a non-existent amp is ignored.

In Enable allows the input path of the Pin Widget to be shut off. The value 1 enables the path.

VRefEn: Voltage Reference Enable controls the VRef signal(s) associated with the Pin Widget. If more than one of the bits in the VRef[7:0] field of the Pin Capabilities parameter (Section 7.3.4.9) are non-zero, then this control allows the signal level to be selected.

The VRefEn field encoding selects one of the possible states for the VRef signal(s). If the value written to this control does not correspond to a supported value as defined in the Pin Capabilities

parameter, the control must either retain the previous value or take the value of 000, which will put the control in a Hi-Z state and prevent damage to any attached components.

Table 82 enumerates the possible values for VRefEn which correlate to the values identified in the Pin Capabilities parameter (see Figure 77).

Table 82. VRefEn Values

VRefEn Encoding	VREF Signal Level
000b	Hi-Z
001b	50%
010b	Ground (0 V)
011b	*Reserved*
100b	80%
101b	100%
110b-111b	*Reserved*

Applies to:

- Pin Complex

7.3.3.14 Unsolicited Response

The **Unsolicited Response** control determines whether the node is enabled to send an unsolicited response, as well as what the tag will be for the response.

This control is only available for nodes which support Unsolicited Responses, as declared in the Function Group Type parameter (Section 7.3.4.4) and the Audio Widget Capabilities parameter (Section 7.3.4.6). The node should be queried to determine if it supports unsolicited responses before getting or setting this control.

Command Options:

Table 83. Connection Select Control

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F08h	0	Bits 31:8 are 0 EnableUnsol is bits 7:0
Set	708h	EnableUnsol is bits 7:0	0

EnableUnsol format:

7	6	5:0
Enable	0	Tag

Figure 64. EnableUnsol Format

Enable controls the actual generation of Unsolicited Responses. If Enable is a 1, Unsolicited Responses may be generated.

Tag is a 6-bit value which is opaque to the codec and is used by software to determine what codec node generated the unsolicited response. The value programmed into the Tag field is returned in the top 6 bits (31:26) of every Unsolicited Response generated by this node.

Applies to:

- All Nodes capable of generating Unsolicited Responses.

7.3.3.15 Pin Sense

The **Pin Sense** control returns the Presence Detect status and the impedance measurement of the device attached to the pin.

Some codecs may require that the impedance measurement be triggered by software; in that case, sending the Execute command will cause the impedance measurement to begin. The "Presence Detect" bit will always be accurate if that functionality is supported by the widget.

Note that the Pin Complex Widget may support the generation of an Unsolicited Response to indicate that the Sense Measurement (either the Presence Detect or the Impedance) value has changed, the generation of which implies that the measurement is complete.

Command Options:

Table 84. Pin Sense

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F09h	0	Presence Detect: bit 31 Impedance Sense: bits 30:0
Execute	709h	Right Chnl: bit 0 Rsvd: bits 1:7	0

Presence Detect is a bit indicating the state of the Presence Detect capability. A 1 indicates that there is "something" plugged into the jack associated with the Pin Complex. This bit will only be valid if the widget has Presence Detect capability as indicated by the "Presence Detect Capable" bit of the Pin Capabilities parameter (see Section 7.3.4.9).

Impedance returns the measured impedance of the widget. A returned value of 0x7FFF,FFFF (all 1's) indicates that a valid sense reading is not available, or the sense measurement is busy (refer to Section 7.3.1) if it has been recently triggered. This field is only valid if the widget has Sense capability as indicated by the "Impedance Sense Capable" bit of the Pin Capabilities parameter.

Right Chnl: Normally impedance sensing is done on the left channel or "tip" of the connector. However, Pin Widgets may optionally support sensing on the right channel or "ring" of the connector. When this bit is 1, the impedance value is taken on the right channel if the Pin Widget supports this; if not supported, this bit is ignored. When this bit is 0, the left channel is sensed.

Applies to:

- Pin Complex

7.3.3.16 EAPD/BTL Enable

EAPD/BTL Enable controls the EAPD pin and configures Pin Widgets into balanced output mode, when these features are supported. It also configures any widget to swap L and R channels when this feature is supported. When this control is referenced to a non-Pin Widget, bits 1 and 0 are not valid, and are considered reserved.

Command Options:

Table 85. EAPD/BTL Enable

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F0Ch	0	bits 31:3 are *Reserved* bit 2 = L-R Swap bit 1 = EAPD bit 0 = BTL
Set	70Ch	bits 7:3 are *Reserved* bit 2 = L-R Swap bit 1 is EAPD bit 0 is BTL	0

L-R Swap causes the left and right channels of the Widget to be swapped for both input and output paths if they exist. The value 1 enables swapping.

EAPD value is reflected on the EAPD pin; a 1 causes the external amplifier to power up, and a 0 causes it to power down. When the EAPD value = 1, the EAPD pin must be placed in a state appropriate to the current power state of the associated Pin Widget; i.e., the external amplifier must be powered down whenever the associated Pin Widget is put in a low power state, even though the EAPD value may remain 1. It is possible that more than one Pin Widget supports the EAPD function as indicated in bit 16 of the Pin Capabilities Parameter (Section 7.3.4.9); this would be true, for example, when external amps were supported for 4-channel output. In this case, each supporting Pin Widget must respond to this control; however, since there is only a single EAPD Pin, there is also only one logical value to set/get, and that value must be accessible, using this control, via any/all supporting Pin Widgets.

BTL controls the output configuration of a Pin Widget which has indicated support for balanced I/O (bit 6, Pin Capabilities Parameter). When this bit is 0, the output drivers are configured in normal, single-ended mode; when this bit is 1, they are configured in balanced mode. Note that in balanced mode, the Pin Widget has twice as many pins as it does in normal mode; i.e., a stereo Pin Widget in balanced mode has four signal pins (in addition to Vref pins). However, in both modes it must appear to software as a single Pin Widget. The additional pins must be reserved to this purpose; thus, in a configuration supporting BTL outputs the additional pins may not be enumerated as a separate Pin Widget.

Applies to:

- Any Audio Widget

7.3.3.17 GPI Data

GPI Data returns the current GPI pin status as a bit field.

Command Options:

Table 86. GPI Data

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F10h	0	Bits 31:8 are *Reserved* Bits 7:0 is Data
Set	710h	Data	0

Data is a bit field reporting the current state of the GPI signals.

Bits in the GPI control are not directly writable, but if the bit is configured as "Sticky" in the GPI Sticky control (below), writing a 1 to the bit will clear the bit.

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group

7.3.3.18 GPI Wake Enable Mask

GPI Wake controls the ability of a change on a GPI pin to cause a Status Change event on the Link.

Command Options:

Table 87. GPI Wake Mask

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F11h	0	Bits 31:8 are *Reserved* Bits 7:0 is Wake
Set	711h	Wake	0

Wake is a bit field representing the state of the GPI Wake bits.

The GPI Wake provides a mask for determining if a GPI change will generate a wake-up (0 = No, 1 = Yes). When the Link is powered down (**RST#** is asserted), a wake-up event will trigger a Status Change Request event on the link. When the Link is powered up, wake events would be Unsolicited Responses, the generation of which is separately controlled (see Section 7.3.3.19).

The default value after cold or register reset for this register (0000h) defaults to all 0's specifying no wake-up event. Non-implemented GPI pins always return 0's.

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group

7.3.3.19 GPI Unsolicited Enable Mask

GPI Unsolicited controls the ability of a change on a GPI pin to cause an Unsolicited Response on the Link.

Command Options:

Table 88. GPI Unsolicited Enable

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F12h	0	Bits 31:8 are *Reserved* Bits 7:0 is UnsolEnable
Set	712h	UnsolEnable	0

UnsolEnable is a bit field representing the state of the GPI Wake bits.

The GPI Unsolicited control provides a mask for determining if a change on a GPI line will generate a wake-up (0 = No, 1 = Yes). If enabled, and the Unsolicited Response control has been set to enable unsolicited responses on the function group, an unsolicited response will be sent when a GPI line changes state.

The default value after cold or register reset for this register (0000h) defaults to all 0's specifying no wake-up event. Non-implemented GPI pins always return 0's.

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group

7.3.3.20 GPI Sticky Mask

GPI Data returns the current GPI pin status as a bit field.

Command Options:

Table 89. GPI Sticky Mask

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F13h	0	Data
Set	713h	Bits 31:8 are *Reserved* Bits 7:0 is Data	0

Data is a bit field reporting the current state of the GPI signals.

The GPI Pin Sticky is a read/write register that defines GPI Input Type (0 = Non-Sticky, 1 = Sticky). GPI inputs configured as Sticky are cleared by writing a 0 to the corresponding bit of the GPI Data control (see Section 7.3.3.17) and by reset.

The default value after cold or register reset for this register (0000h) is all pins Non-Sticky. Unimplemented GPI pins always return 0s. Sticky is defined as edge-sensitive, Non-Sticky as Level-sensitive.

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group

7.3.3.21 GPO Data

GPO Data controls the state of the GPO pins.

Command Options:

Table 90. GPO Data

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F14h	0	Data
Set	714h	Bits 31:8 are *Reserved* Bits 7:0 is Data	0

Data is a bit field representing the state of the GPO signals on the codec.

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group

7.3.3.22 GPIO Data

GPIO Data sets and returns the data associated with the GPIO signals. The control is a bit field, with each bit corresponding to a GPIO pin starting from bit 0.

Command Options:

Table 91. GPIO Data

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F15h	0	Data
Set	715h	Bits 31:8 are *Reserved* Bits 7:0 is Data	0

Data is a bit field representing the current state of the GPIO signals on the codec.

For bits in the GPIO Data control that have corresponding bits in the GPIO Direction control set to 0, the pin behaves as an input and the bit value will be a read-only value indicating the state being sensed by the pin. Bits that are configured as outputs in the GPIO Data control are read/write in the GPIO Data control, and the value written will determine the value driven on the pin. Reads from output bits will reflect the value being driven by the GPIO pin.

Note that if the corresponding bit in the GPIO Enable control is not set, pins configured as outputs will not drive associated bit value (as the pin must be in a Hi-Z state), but the value returned on a read will still reflect the value that would be driven if the pin were to be enabled in the GPIO Enable control.

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group

7.3.3.23 GPIO Enable Mask

GPIO Enable controls whether a GPIO pin is enabled or not. The control is a bit field, with each bit corresponding to a GPIO pin starting from bit 0.

Command Options:

Table 92. GPIO Enable

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F16h	0	Enable
Set	716h	Bits 31:8 are *Reserved* Bits 7:0 is Enable	0

Enable is a bit field representing the configuration of the GPIO signals. For each GPIO pin, there is an associated bit in the Enable field. If the bit associated with a pin is 0, the pin is disabled, and must be in a Hi-Z state. If the bit is a 1, the GPIO pin is enabled and the pin's behavior will be determined by the GPIO Direction control.

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group

7.3.3.24 GPIO Direction

GPIO Direction determines whether a given GPIO is configured to drive or sense the connected signal. The control is a bit field, with each bit corresponding to a GPIO pin starting from bit 0.

Command Options:

Table 93. GPIO Direction

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F17h	0	Control
Set	717h	Bits 31:8 are *Reserved* Bits 7:0 is Control	0

Control is a bit field representing the configuration of the GPIO signals. If a bit position is a 0, the associated GPIO signal is configured as an input. If the bit is set to a 1, the associated GPIO signal is configured as an output.

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group

7.3.3.25 GPIO Wake Enable Mask

GPIO Wake controls the ability of a change on a GPIO pin configured as an input to cause a Status Change event on the Link.

Command Options:

Table 94. GPIO Wake Enable

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F18h	0	Bits 31:8 are *Reserved* Bits 7:0 is Wake
Set	718h	Wake	0

Wake is a bit field representing the state of the GPIO Wake bits.

The GPIO Wake provides a mask for determining if a change on a GPIO pin will generate a wake-up (0 = No, 1 = Yes). When the Link is powered down (**RST#** is asserted), a wake-up event will trigger a Status Change Request event on the link. When the Link is powered up, wake events would be Unsolicited Responses, the generation of which is separately controlled (see Section 7.3.3.26).

The default value after cold or register reset for this register (0000h) defaults to all 0's specifying no wake-up event. Non-implemented GPIO pins always return 0's.

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group

7.3.3.26 GPIO Unsolicited Enable Mask

GPIO Unsolicited controls the ability of a change on a GPIO pin configured as an input to cause an Unsolicited Response on the Link.

Command Options:

Table 95. GPIO Unsolicited Enable

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F19h	0	Bits 31:8 are *Reserved* Bits 7:0 is UnsolEnable
Set	719h	UnsolEnable	0

UnsolEnable is a bit field representing the state of the GPIO Unsolicited Enable bits.

The GPIO Unsolicited Enable Mask control provides a mask for determining if a change on a GPIO line will generate a wake-up (0 = No, 1 = Yes). If enabled, and the Unsolicited Response control has been set to enable unsolicited responses on the widget, an unsolicited response will be sent when a GPIO line changes state.

The default value after cold or register reset for this register (0000h) defaults to all 0's specifying no wake-up event. Non-implemented GPIO pins always return 0's.

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group

7.3.3.27 GPIO Sticky Mask

GPIO Data returns the current GPIO pin status as a bit field.

Command Options:

Table 96. GPIO Sticky Mask

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F1Ah	0	Data
Set	71Ah	Bits 31:8 are *Reserved* Bits 7:0 is Data	0

Data is a bit field reporting the current state of the GPIO signals.

The GPIO Sticky Mask defines GPIO Input Type (0 = Non-Sticky, 1 = Sticky) when a GPIO pin is configured as an input. GPIO inputs configured as Sticky are cleared by writing a 0 to the corresponding bit of the GPIO Data Control (see Section 7.3.3.22) and by reset.

The default value after cold or register reset for this register (0000h) is all pins Non-Sticky. Unimplemented GPIO pins always return 0's. Sticky is defined as Edge sensitive, Non-Sticky as Level-sensitive.

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group

7.3.3.28 Beep Generation

The **Beep Generation** command causes the Beep Generator Widget to beep or to stop beeping. The intended use of this command is for BIOS POST beeps, not for generating high quality audio output. Note that the Beep Generator Widget may have an optional output amplifier as defined in the "Audio Widget Capabilities" parameter (Section 7.3.4.6). If this amp is present, it is controlled with the normal amplifier controls.

Command Options:

Table 97. Beep Generation

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F0Ah	0	Bits 31:8 are *Reserved* Bits 7:0 are Divider
Set	70Ah	Bits 7:0 are Divider	0

Divider indicates the divisor for the 48-kHz link frame tick used to generate the beep.

The beep frequency generated is the result of dividing the 48-kHz link frame clock by 4 times the number specified in Divider, allowing tones from 47 Hz to 12 kHz. When a non-zero data value is sent to enable the Beep, the codec generates the beep tone on all Pin Complexes that are currently configured as outputs. The codec may either disable any normal streaming on the output pins, or it may mix the beep with the active output streams depending on the design of the codec.

A value of 00h in bits Divider disables internal PC Beep generation and enables normal operation of the codec.

The generated signal is not intended to be a high quality sine wave. The clock output rounded with a capacitor provides sufficient signal quality to provide beep code signaling.

Applies to:

- Beep Generator Widget

7.3.3.29 Volume Knob

Volume Knob provides the controls for an optional external hardware volume control.

Command Options:

Table 98. Volume Knob Control

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F0Fh	0	Bit 7 is Direct Bits 6:0 is Volume
Set	70Fh	Bit 7 is Direct Bits 6:0 is Volume	0

Direct causes the Volume Control Widget to directly control the hardware volume of the slave amplifiers. If "Direct" is set to a 0, the volume control will not directly affect the volume of the slaves amplifiers; rather, the software receives an unsolicited response, reads the volume control, and then programs the appropriate amplifiers correctly.

Volume is specified in steps, as is amplifier gain. If two amplifiers slaved to the Volume Knob control have different "StepSize" parameters, they are both adjusted by the same number of steps, implying a differing total dB treatment. If the Volume Knob control has more steps than a slave amplifier is capable of supporting (as indicated in the Volume Knob Capabilities parameter), the amplifier remains at its limiting value.

Applies to:

- Volume Knob Widget

7.3.3.30 Subsystem ID

This set of controls provides read/write access to the 32-bit Subsystem ID register contained in each Functional Group. This register is used to identify the functional group to the software PnP subsystem. The Assembly ID (8 bits) is intended primarily for modems; when used, its value is loaded from a "strapping option" or other board-specific mechanism at power-up time. The Subsystem ID (24 bits) is intended to be "hardwired" into the silicon. The silicon vendor is responsible for defining the values used in both fields. It is recommended that this control default to a non-zero value.

System BIOS or other means may also be used to write to this register to set it. In such cases, the register should be set to its proper value at all times the operating system or application software may read the register. The ability for software to write this register is not a requirement as long as operating system requirements for unique Subsystem ID's can be met through other means.

In the case where the SSID is determined through external means, for instance the codec reading an external EEPROM to load register defaults, this register may return a value of 0xFFFFFFFF for up to 7 ms after the de-assertion of the Link **RST#** signal before changing to reflect the proper value.

31:8	7:0
Subsystem ID	Assembly ID

Figure 65. Subsystem ID Register

Command Options:

Table 99. Subsystem ID

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F20h[6]	0	Subsystem ID bits [31:0]
Set 1	720h	Subsystem ID bits [7:0]	0
Set 2	721h	Subsystem ID bits [15:8]	0
Set 3	722h	Subsystem ID bits [23:16]	0
Set 4	723h	Subsystem ID bits [31:24]	0

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group

7.3.3.31 Configuration Default

The **Configuration Default** is a 32-bit register required in each Pin Widget. It is used by software as an aid in determining the configuration of jacks and devices attached to the codec. At the time the codec is first powered on, this register is internally loaded with default values indicating the typical system use of this particular pin/jack. After this initial loading, it is completely codec opaque, and its state, including any software writes into the register, must be preserved across reset events. Its state need not be preserved across power level changes.

[6] The Verb Codes F21h, F22h, and F23h are reserved for the Subsystem ID register and must not be reassigned to anything else. However, they need not be implemented since standard software drivers will not use them. If a codec elects to respond to these codes, the response must be identical in all respects to the response to Verb Code F20h.

Command Options:

Table 100. Configuration Default

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F1Ch[7]	0	Config bits [31:0]
Set 1	71Ch	Config bits [7:0]	0
Set 2	71Dh	Config bits [15:8]	0
Set 3	71Eh	Config bits [23:16]	0
Set 4	71Fh	Config bits [31:24]	0

Data Structure:

The Configuration Default register is defined as shown in Figure 66.

31:30	29:24	23:20	19:16	15:12	11:8	7:4	3:0
Port Connectivity	Location	Default Device	Connection Type	Color	Misc	Default Association	Sequence

Figure 66. Configuration Data Structure

Port Connectivity[1:0] indicates the external connectivity of the Pin Complex. Software can use this value to know what Pin Complexes are connected to jacks, internal devices, or not connected at all. The encodings of the Port Connectivity field are defined in Table 101.

Location[5:0] indicates the physical location of the jack or device to which the pin complex is connected. This allows software to indicate, for instance, that the device is the "Front Panel Headphone Jack" as opposed to rear panel connections. The encodings of the Location field are defined in Table 102.

The Location field is divided into two pieces, the upper bits [5:4] and the lower bits [3:0]. The upper bits [5:4] provide a gross location, such as "External" (on the primary system chassis, accessible to the user), "Internal" (on the motherboard, not accessible without opening the box), on a separate chassis (such as a mobile box), or other.

The lower bits [3:0] provide a geometric location, such as "Front," "Left," etc., or provide special encodings to indicate locations such as mobile lid mounted microphones. An "x" in Table 102 indicates a combination that software should support. While all combinations are permitted, they are not all likely or expected.

Default Device[3:0] indicates the intended use of the jack or device. This can indicate either the label on the jack or the device that is hardwired to the port, as with integrated speakers and the like. The encodings of the Default Device field are defined in Table 103.

[7] The Verb Codes F1Dh, F1Eh, and F1Fh are reserved for the Configuration Default register and must not be reassigned to anything else. However, they need not be implemented since standard software drivers will not use them. If a codec elects to respond to these codes, the response must be identical in all respects to the response to Verb Code F1Ch.

Connection Type[3:0] indicates the type of physical connection, such as a 1/8-inch stereo jack or an optical digital connector, etc. Software can use this information to provide helpful user interface descriptions to the user or to modify reported codec capabilities based on the capabilities of the physical transport external to the codec. The encodings of the Connection Type field are defined in Table 104.

Color[3:0] indicates the color of the physical jack for use by software. Encodings for the Color field are defined in

Table 105.

Misc[3:0] is a bit field used to indicate other information about the jack. Currently, only bit 0 is defined. If bit 0 is set, it indicates that the jack has no presence detect capability, so even if a Pin Complex indicates that the codec hardware supports the presence detect functionality on the jack, the external circuitry is not capable of supporting the functionality. The bit definitions for the Misc field are in Table 106.

Default Association and **Sequence** are used together by software to group Pin Complexes (and therefore jacks) together into functional blocks to support multichannel operation. Software may assume that all jacks with the same association number are intended to be grouped together, for instance to provide six channel analog output. The Default Association can also be used by software to prioritize resource allocation in constrained situations. Lower Default Association values would be higher in priority for resources such as processing nodes or Input and Output Converters. Note that this is the default association only, and software can override this value if required, in particular if the user provides additional information about the particular system configuration. A value of 0000b is reserved and should not be used. Software may interpret this value to indicate that the Pin Configuration data has not been properly initialized. A value of 1111b is a special value indicating that the Association has the lowest priority. Multiple different Pin Complexes may share this value, and each is intended to be exposed as independent devices.

Sequence indicates the order of the jacks in the association group. The lowest numbered jack in the association group should be assigned the lowest numbered channels in the stream, etc. The numbers need not be sequential within the group, only the order matters. Sequence numbers within a set of Default Associations must be unique.

Table 101. Port Connectivity

Value	Value
00b	The Port Complex is connected to a jack (1/8", ATAPI, etc.).
01b	No physical connection for Port.
10b	A fixed function device (integrated speaker, integrated mic, etc.) is attached.
11b	Both a jack and an internal device are attached. The Information provided in all other fields refers to the integrated device. The PD pin will reflect the status of the jack; the user will need to be queried to figure out what it is.

Table 102. Location

Bits [5:4]	Bits [3:0] 0h: N/A	1h: Rear	2h: Front	3h: Left	4h: Right	5h: Top	6h: Bottom	7h: Special	8h: Special	9h: Special	Ah-Fh: Reserved
00b: External on primary chassis	x	x	x	x	x	x	x	x (Rear panel)	x (Drive bay)		
01b: Internal	x							x (riser)	x (HDMI)	x (ATAPI)	
10b: Separate chassis	x	x	x	x	x	x	x				
11b: Other	x						x	x (Mobile Lid–Inside) (e.g., mic inside mobile lid)	x (Mobile Lid–Outside)		

Table 103. Default Device

Default Device	Encoding
Line Out	0h
Speaker	1h
HP Out	2h
CD	3h
SPDIF Out	4h
Digital Other Out	5h
Modem Line Side	6h
Modem Handset Side	7h
Line In	8h
AUX	9h
Mic In	Ah
Telephony	Bh
SPDIF In	Ch
Digital Other In	Dh
Reserved	Eh
Other	Fh

Table 104. Connection Type

Connection Type	Encoding
Unknown	0h
1/8" stereo/mono	1h
1/4" stereo/mono	2h
ATAPI internal	3h
RCA	4h
Optical	5h
Other Digital	6h
Other Analog	7h
Multichannel Analog (DIN)	8h
XLR/Professional	9h
RJ-11 (Modem)	Ah
Combination	Bh
Other	Fh

Table 105. Color

Color	Encoding
Unknown	0h
Black	1h
Grey	2h
Blue	3h
Green	4h
Red	5h
Orange	6h
Yellow	7h
Purple	8h
Pink	9h
Reserved	A-Dh
White	Eh
Other	Fh

Table 106. Misc

Misc	Bit
Reserved	3
Reserved	2
Reserved	1
Jack Detect Override	0

Applies to:

- Pin Widget

7.3.3.32 Stripe Control

The **Stripe Control** verb reports and controls the strip capability of an Audio Output Converter. This verb needs to be implemented only for an Audio Output Converter, and only if the Stripe bit of the Audio Widget Capabilities parameter is a 1.

Command Options:

Table 107. Stripe Control

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Get	F24h	0	Stripe bits [31:0]
Set 1	720h	Stripe bits [7:0]	0

Data Structure:

The Stripe Control register is defined as shown in Figure 67.

31:23	22:20	7:2	1:0
Reserved	Stripe Capability [2:0]	*Reserved*	Stripe Control [1:0]

Figure 67. Stripe Control Register

- **Stripe Capability[2:0]** is a bit field indicating on which **SDO** lines the Audio Output Converter Widget may receive data from the controller. Bit 0 should always be 1, indicating that the widget can receive data on **SDI**[0]. If bit 1 is set, it indicates that the widget can receive stream data on **SDI**[0:1]. If bit 2 is set, the widget can receive stream data on **SDI**[0:3]. These bits must be dynamically updated by the codec based on which SDI lines have a connection between the codec and the controller. This can be determined by the codec by observing which **SDO** lines toggle when commands are sent, as indicated in Section 5.3.2.3.
- **Stripe Control [1:0]** is an encoded field controlling which **SDO** lines a stream will be transmitted on. An Audio Output Converter Widget uses this information to know which **SDO** lines it should decode the output stream. This field should be programmed by software to match the Stripe Control field of the Stream Descriptor Control (Section 3.3.35).

Applies to:

- Output Converter

7.3.3.33 Function Reset

The **Function Reset** command causes the functional unit, and all widgets associated with the functional unit, to return to their power-on reset values. Note that some controls such as the Configuration Default controls should not be reset with this command. It is also possible that certain other controls, such as Caller-ID, should not be reset.

This command does not affect the Link interface logic, which must be reset with the link **RST#** signal. Therefore, a codec must not initiate a Status Change request on the link.

When a codec receives the Function reset verb, the expected behavior is that the codec will issue a response to the verb to acknowledge receipt, and then reset the affected Function Group controls. The codec must be ready to respond to the verbs on the Link frame after the frame on which it returns its response to the Reset command.

Command Options:

Table 108. Function Reset

	Verb ID	Payload (8 Bits)	Response (32 Bits)
Execute	7FFh	0	0

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group

7.3.4 Parameters

Parameters are static, Read Only values in the codec used to convey codec, function, and node capabilities to the software.

7.3.4.1 Vendor ID

Vendor ID returns the Device and Vendor ID values for the codec. These are each 16-bit values used to identify the codec to the PnP subsystem. Note that this is the first parameter read in enumerating a codec (root node) and as such will be used to determine the possible presence of multiple SDI signals (refer to Section 7.3.2). In the case of multiple SDIs with a single root node, the response to this parameter fetch must always be returned on the SDI which the codec designates as "primary," regardless of which CAd the request was generated on.

Parameter ID: 00h

Response Format:

31:16	15:0
Vendor ID	Device ID

Figure 68. Vendor ID Response Format

Applies to:

- Root Node

7.3.4.2 Revision ID

The **Revision ID** parameter returns the codec revision ID value for the codec. This is an 8-bit value used to identify the codec to the PnP subsystem.

Parameter ID: 02h

Response Format:

31:24	23:20	19:16	15:8	7:0
Rsvd	MajRev	MinRev	Revision ID	Stepping ID

Figure 69. Revision ID Response Format

MajRev (4 bits) is the major revision number (left of the decimal) of the High Definition Audio Specification to which the codec is fully compliant.

MinRev (4 bits) is the minor revision number (right of the decimal) or "dot number" of the High Definition Audio Specification to which the codec is fully compliant.

Revision ID (8 bits) is the vendor's revision number for this given Device ID.

Stepping ID (8 bits) is an optional vendor stepping number within the given Revision ID.

Applies to:

- Root Node

7.3.4.3 Subordinate Node Count

The **Subordinate Node Count** parameter provides information about the function group nodes associated with the codec (root node), as well as the widget nodes associated with a function group.

For a Root Node, the "Total Number of Nodes" parameter is the number of Function Group nodes in the codec, the starting address of which is provided by the "Starting Node Number" parameter.

For a Function Group node, the "Total Number of Nodes" parameter is the number of widget or functional nodes in the Functional Group, the starting address of which is provided by the "Starting Node Number" parameter.

Parameter ID: 04h

Response Format:

31:24	23:16	15:8	7:0
Reserved	Starting Node Number	*Reserved*	Total Number of Nodes

Figure 70. Subordinate Node Count Response Format

Applies to:

- Root Node
- Audio Function
- Modem Function Group
- Other Function Group

7.3.4.4 Function Group Type

The **Function Group Type** parameter returns a value describing what the type of node is being addressed. This parameter is primarily useful for identifying the type of Function Group a node represents (such as Audio versus Modem) but can also be used to identify the type of "Other" or vendor specific nodes.

Parameter ID: 05h

Response Format:

31:9	8	7:0
Reserved	UnSol Capable	NodeType

Figure 71. Function Group Type Response Format

UnSol Capable indicates (when = 1) that this node is capable of generating an unsolicited response and will respond to the Unsolicited Response verb (Section 7.3.3.14). Note that if this node does not generate unsolicited responses, subordinate nodes (widgets) may still do so.

Table 109. Node Type

Node Type	Value
00h	*Reserved*
01h	Audio Function Group
02h	Vendor Defined Modem Function Group
03-7Fh	*Reserved*
80-FFh	Vendor defined Function Group

As shown, the upper 128 type encodings may be used to identify vendor specific nodes other than audio widgets. Vendor specific Audio Widgets – i.e., those vendor defined widgets that are enumerated hierarchically within an Audio Function Group – do not have this parameter, but must be identified as a "Vendor Defined Audio Widget" (type = Fh) in the "Audio Widget Capabilities" parameter (see Section 7.3.4.6), since their type will be queried by the audio function driver using that parameter.

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Groups
- Vendor Defined Widgets outside an Audio Function Group

7.3.4.5 Audio Function Group Capabilities

The **Audio Parameter** returns a set of bit fields describing the audio capabilities of the Audio Function.

Parameter ID: 08h

Response Format:

31:17	16	15:12	11:8	7:4	3:0
Rsvd	Beep Gen	*Rsvd*	Input Delay	*Rsvd*	Output Delay

Figure 72. Audio Function Group Capabilities Response Format

BeepGen indicates the presence of an optional Beep Generator with this Audio Function Group (refer to Section 7.2.3.8).

Input Delay is a 4-bit value representing the number of samples between when the sample is received as an analog signal at the pin and when the digital representation is transmitted on the High Definition Audio Link. This may be a "typical" value. If this is 0, the widgets along the critical path should be queried, and each individual widget must report its individual delay.

Output Delay is a four bit value representing the number of samples between when the sample is received from the Link and when it appears as an analog signal at the pin. This may be a "typical" value. If this is 0, the widgets along the critical path should be queried, and each individual widget must report its individual delay.

Applies to:

- Audio Function Group

7.3.4.6 Audio Widget Capabilities

The Audio Capabilities control returns a set of bit fields describing the audio capabilities of the widget.

Parameter ID: 09h

Response Format:

31:24	23:20	19:16	15:12	11	10	9	8	7	6	5	4	3	2	1	0
Rsvd	Type	Delay	*Rsvd*	L-R Swap	Power Cntrl	Digital	Conn List	Unsol Cap-able	Proc Widget	Stripe	Format Override	Amp Param Over-ride	Out Amp Present	In Amp Present	Stereo

Figure 73. Audio Widget Capabilities Response Format

Type defines the functionality of the widget node. This is an enumerated list.

Table 110. Widget Type

Value	Type
0h	Audio Output
1h	Audio Input
2h	Audio Mixer
3h	Audio Selector
4h	Pin Complex
5h	Power Widget
6h	Volume Knob Widget[8]
7h	Beep Generator Widget[9]
8h-Eh	*Reserved*
Fh	Vendor defined audio widget

Any vendor defined widget that is enumerated hierarchically within an Audio Function Group must be identified as a vendor defined type (Fh) using this parameter.

Delay indicates the number of sample delays through the widget. This may be 0 if the delay value in the Audio Function Parameters is supplied to represent the entire path.

L-R Swap indicates the capability of swapping the left and right channels through the Audio Widget. If the Audio Widget is both input and output capable (e.g., Pin Widget), then swapping must be supported for both input and output paths. Default (0) is no swap capability.

PowerCntrl indicates that the Power State control is supported on this widget. This allows finer grained power management than just at the Function Group level for widgets which support it.

Digital indicates that a widget supports a digital stream. If the bit is a 1, it is a digital widget. For an Input or Output converter, for instance, this means the widget is translating between the High Definition Audio Link and a digital format such as S/P-DIF or I2S rather than analog data.

ConnList indicates whether a connection list is present on the widget. If the bit is a 1, the Connection List Length parameter and the Connection List Entry controls should be queried to discover the input connections. This bit must be a 1 for Input Converters, Sum Widgets, and Selector Widgets. The bit may be a 0 for Output Converters if the only connection for the widget is to the High Definition Audio Link.

If **Unsol Capable** is a 1, the audio widget supports unsolicited responses. The Unsolicited Response command can be used to configure and enable Unsolicited Response generation. When this parameter is associated with a Pin Widget, then setting this bit requires that the Pin Widget generate an unsolicited response (when enabled) whenever the "Presence Detect" bit (see Section 7.3.3.15) changes state.

If **ProcWidget** is a 1, the "Processing Controls" parameter should be queried for more information about the widget's processing controls.

[8] In the case of the Volume Knob Widget, none of the parameter bits [19:0] are used and may be omitted or set to 0. However, software assumes the capability of unsolicited responses and a connection list, as these are required by this widget type.

[9] In the case of the Beep Generator Widget, the only meaningful parameter bits are 2 ("Out Amp Present") and 3 ("Amp Param Override"). None of the other parameter bits are used and may be omitted or set to 0.

The **Stripe** bit defines whether the Audio Output Converter Widget supports striping, as defined in Section 5.3.2.3. If Stripe is a 1, the Audio Output Converter Widget must support the Stripe Control verb. Stripe is only defined for Audio Output Converter Widgets; for all other widget types, this bit must be 0.

If **Format Override** is a 1, the widget contains format information, and the "Supported Formats" and "Supported PCM Bits, Rates" should be queried for the widget's format capabilities. If this bit is a 0, then the Audio Function node must contain default amplifier parameters, and that node's format related parameters should be queried to determine the format parameters. This bit is not applicable to, and is always 0 for, Pin Complex Widgets.

If **Amp Param Override** is a 1, the widget contains its own amplifier parameters. If this bit is a 0, then the Audio Function node must contain default amplifier parameters, and they should be used to define all amplifier parameters (both input and output) in this widget.

If **(In|Out) AmpPresent** is a 1, the widget contains an input or output amplifier, as indicated. The Amp Param Override bit should be examined to determine whether the widget contains default amplifier parameters or has amplifier parameters that need to be explicitly queried. The "In Amp Present" bit is only relevant for Sum Widgets, Input Converters, and Pin Complexes. The "Out Amp Present" bit is only relevant for Selector Widgets, Sum Widgets, Output Converter Widgets, and Pin Complex Widgets.

The **Stereo** bit determines if the widget is a stereo or mono widget. If the "Stereo" bit is a 1, the widget is a stereo widget.

Applies to:

- Input Converter Widget
- Output Converter Widget
- Selector Widget
- Mixer Widget
- Pin Widget

7.3.4.7 Supported PCM Size, Rates

The Supported Rates parameter returns a bit field describing the bit depth and sample rate capabilities of the widget when PCM formatted data is being rendered or captured.

Parameter ID: 0Ah

Response Format:

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
rsvd	*rsvd*	*rsvd*	*rsvd*	*rsvd*	*rsvd*	*rsvd*	*rsvd*	*rsvd*	*rsvd*	*rsvd*	B32	B24	B20	B16	B8
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
rsvd	*rsvd*	*rsvd*	*rsvd*	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1

Figure 74. Supported PCM Size, Rates Return Format

Table 111. Bit Depth and Sample Rate

Bit	Meaning	
B32	32 bit audio formats are supported	
B24	24 bit audio formats are supported	
B20	20 bit audio formats are supported	
B16	16 bit audio formats are supported	
B8	8 bit audio formats are supported	
	Rate (kHz)	**Mult/Div**
R1	8.0	1/6 * 48
R2	11.025	1/4 * 44.1
R3	16.0	1/3 * 48
R4	22.05	1/2 * 44.1
R5	32.0	2/3 * 48
R6	44.1	
R7	48.0	(Must be supported by all codecs)
R8	88.2	2/1 * 44.1
R9	96.0	2/1 * 48
R10	176.4	4/1 * 44.1
R11	192.0	4/1 * 48
R12	384	8/1 * 48

Applies to:

- Audio Function Group (as default for all widgets within the Audio Function)
- Audio Input Converter
- Audio Output Converter

7.3.4.8 Supported Stream Formats

The Supported Stream Formats parameter returns a bit field describing the format capabilities of the widget.

Parameter ID: 0Bh

Response Format:

31:3	2	1	0
rsvd	AC3	Float32	PCM

Figure 75. Supported Stream Formats Response Format

The **PCM** bit set indicates that the widget supports PCM formatted data. The PCM formats supported are determined using the "Supported PCM Bits, Rates" parameter.

The **Float32** bit indicates that Float32 formatted data is supported. The "Supported PCM Bits, Rates" determine the sample rates, but only 32-bit data is supported.

The **AC3** bit indicates that Dolby* AC-3 formatted data is supported. The "Supported PCM Bits, Rates" parameter determines the sample rates, but only 16-bit data is supported.

Applies to:

- Audio Function Group (as default for all widgets within the Audio Function)
- Audio Input Converter
- Audio Output Converter

7.3.4.9 Pin Capabilities

The Pin Capabilities parameter returns a bit field describing the capabilities of the Pin Complex Widget.

Parameter ID: 0Ch

Response Format:

31:17	16	15:8	7	6	5	4	3	2	1	0
Rsvd	EAPD Capable	VRef Control	*Rsvd*	Balanced I/O Pins	Input Capable	Output Capable	Headphone Drive Capable	Presence Detect Capable	Trigger Req'd	Impedance Sense Capable

Figure 76. Pin Capabilities Response Format

EAPD Capable indicates the codec has an EAPD pin and that this Pin Widget provides support for controlling that pin.

VRef Control[7:0] is a bit field used to indicate what voltages may be produced on the associated VRef pin(s). If all bits in the bit field are 0, then VRef generation is not supported by the Pin Complex. Also, if the Input Capable bit is a 0, then the VRef bit field has no meaning and all bits must be 0.

If the Output Capable bit and any bits in the VRef field are set, then bit 0 (Hi-Z) must also be set to indicate that the VRef signal can be turned off to support output devices.

Figure 77 describes the VRef bit field. A 1 in any position indicates that the associated signal level is supported. All values of VRef are specified as a percentage of the analog voltage rail, AVdd.

7:6	5	4	3	2	1	0
Rsvd	100%	80%	*Rsvd*	Ground	50%	Hi-Z

Figure 77. VRef Bit Field

Balanced I/O Pins indicates that the Pin Complex Widget has balanced pins.

Input Capable indicates whether the pin complex supports input. If Input Capable is a 1, the pin is capable of input.

Output Capable indicates whether the pin complex supports output. If Output Capable is a 1, the pin is capable of output.

Headphone Drive Capable indicates that the pin has an amplifier with sufficient current drive to drive headphones. If Output Capable is a 0, then this bit has no meaning and must be 0.

Presence Detect Capable indicates whether the pin complex can perform presence detect to determine whether there is anything plugged in. Presence detect does not indicate *what* is plugged in, only that *something* is plugged in.

Trigger Required indicates whether a trigger is required for an impedance measurement (see Section 7.3.3.15).

Impedance Sense Capable indicates whether the pin complex supports impedance sense on the attached peripheral to determine what it is. More accurate (possibly sequenced) forms of peripheral discrimination may be supported independent of this capability; however, if this bit is a 1, then the codec must support at least the basic impedance test as described in Section 7.3.3.15.

Applies to:

- Pin Widget

7.3.4.10 Amplifier Capabilities

The "Amplifier Properties" parameters return the parameters for the input or the output amplifier on a node. In the case of a Pin Widget, the terms input and output are relative to the codec itself; for all other widgets, these terms are relative to the node. The amplifier capabilities are indicated by the step size of the amplifier, the number of steps, the offset of the range with respect to 0 dB, and whether the amplifier supports mute.

Parameter ID: 0Dh for Input amplifiers; 12h for Output amplifiers

Response Format:

31	30:23	22:16	15	14:8	7	6:0
Mute Capable	*Rsvd*	StepSize	*Rsvd*	NumSteps	*Rsvd*	Offset

Figure 78. Amplifier Capabilities Response Format

StepSize (7 bits) indicates the size of each step in the gain range. Each individual step may be 0-32 dB specified in 0.25-dB steps. A value of 0 indicates 0.25-dB steps, while a value of 127d indicates a 32-dB step.

NumSteps (7 bits) indicates the number of steps in the gain range. There may be from 1 to 128 steps in the amplifier gain range. (0d means 1 step, 127d means 128 steps). A value of 0 (1 step) means that the gain is fixed and may not be changed.

Offset (7 bits) indicates which step is 0 dB. If there are two or more steps, one of the step values must correspond to a value of 0 dB. The "Offset" value reflects the value which, if programmed in to the Amplifier Gain control, would result in a gain of 0 dB.

Mute Capable (1 bit) reports if the respective amplifier is capable of muting. Muting implies a –infinity gain (no sound passes), but the actual performance is determined by the hardware.

Applies to:

- Audio Function Group (as default for all widgets within the Audio Function)
- Audio Input Converter
- Audio Output Converter
- Mixer Widget
- Selector Widget
- Pin Widget
- Other Widget

7.3.4.11 Connection List Length

Returns the length of the connection list for the widget.

Parameter ID: 0Eh

Response Format:

31:8	7	6:0
Reserved	Long Form	Connection List Length

Figure 79. Connection List Length Response Format

Long Form indicates whether the items in the connection list are long form or short form as described in Section 7.1.2.

Connection List Length is an integer indicating the number of items in the connection list. If Connection List Length is 1, there is only one hard-wired input possible, which is read from the Connection List, and there is no Connection Select Control (see Section 7.3.3.2).

Applies to:

- Audio Input Converter
- Mixer Widget
- Selector Widget
- Pin Widget
- Power Widget

7.3.4.12 Supported Power States

Returns a bit field describing the power states supported by the functional unit and widgets.

Parameter ID: 0Fh

Response Format:

31:8	3	2	1	0
Reserved	D3Sup	D2Sup	D1Sup	D0Sup

Figure 80. Supported Power States Response Format

Applies to:

- Audio Function
- Modem Function
- Other Functions
- Power Widget
- Other Widgets (optional)

7.3.4.13 Processing Capabilities

Parameter ID: 10h

Response Format:

31:8	15:8	7:1	0
Reserved	NumCoeff	0	Benign

Figure 81. Processing Capabilities Response Format

Benign will be a 1 if the processing on the widget can be placed in a mode which is linear and time invariant, such as equalization or speaker compensation processing. The Processing control can cause the processing to enter this state.

NumCoeff will indicate the number of coefficients to be loaded on the widget if the processing widget supports loadable parameters. If loadable coefficients are not supported, this value must be 0.

Applies to:

- Input Converter
- Output Converter
- Selector Widget
- Pin Complex
- Other Widget

7.3.4.14 GP I/O Count

Parameter ID: 11h

Response Format:

31	30	29:24	23:16	15:8	7:0
GPIWake	*GPIUnsol*	*Reserved*	NumGPIs	NumGPOs	NumGPIOs

Figure 82. GP I/O Capabilities Response Format

GPIOWake will be reported as 1 if the GPIs and GPIOs configured as inputs can cause a wake when there is a change in level on the pin by generating a Status Change event on the link when the link **RST#** is enabled. This capability must be enabled using the "GPI Wake Enable Mask" or 'GPIO Wake Enable Mask' controls.

GPIOUnsol will be reported as 1 if the GPIs and GPIOs configured as inputs can cause an Unsolicited Response to be generated when there is a change in level on the pin. This capability must be enabled using the "GPI Unsolicited Enable Mask" or "GPIO Unsolicited Enable Mask" controls.

NumGPIs is an integer representing the number of GPI pins supported by the function. There may be from 0 to 7 GPI bits in the function.

NumGPOs is an integer representing the number of GPOs supported by the function. There may be from 0 to 7 GPO bits in the function.

NumGPIOs is an integer representing the number of GPIOs supported by the function. There may be from 0 to 7 GPIO bits in the function.

Applies to:

- Audio Function Group
- Modem Function Group
- Other Function Group

7.3.4.15 Volume Knob Capabilities

Parameter ID: 13h

Response Format:

31:8	7	6:0
Reserved	Delta	NumSteps

Figure 83. Volume Knob Capabilities Response Format

Delta indicates if software can write a base volume to the Volume Control Knob. If this bit is a 1, software can write to the volume control; any volume changes in HW from that point are a delta from the written value. If this bit is a 0, then the volume is absolute (e.g., from a pot) and software cannot modify the value.

NumSteps is the total number of steps in the range of the volume knob. This is similar to the NumSteps parameter of amplifiers, but there is not a StepSize specified.

Applies to:

- Volume Knob Widget

7.3.5 Vendor Defined Verbs

Codec vendors may choose to use additional verbs not defined in this specification to provide additional functionality. A vendor may not use a vendor-defined verb to provide functionality that a specification defined verb already provides; the specification defined verb should always be used whenever possible.

A vendor may use verb encodings of F70 through FFF to implement vendor defined verbs on defined Audio Widget types.

On vendor defined widget types, such as modem widgets, any verb encoding that does not conflict with required verbs, such as the "Get Parameter" verb, may be used.

7.3.6 Required Parameter and Control Support

Table 112 specifies which parameters are required (R) for each specification-defined node. It also indicates optional (o) parameters which are used to declare the presence of optional features in the associated node. A shaded square in the table indicates that the subject parameter is not applicable to the subject node type. The squares marked with (a) indicate an alternative; the parameter is required in either the Audio Function Group (AFG), to be used as a default, or else in all of the indicated widgets. If these parameters are present in the AFG, they are only needed in the individual widgets that have non-default capabilities. Some parameters are marked with an asterisk (*) for the "Vendor_Specific_Audio_Widget indicating they are not required by the specification since a vendor specific node may largely define its own parameters. If, however, the vendor specific node implements features that can be defined by an existing parameter, then using the standard parameter is preferable to defining a new one.

Table 112. Required Support for Parameters

Required Parameter Support	Parameter ID	Root Node	Audio Function Group	Modem Function Group	Vendor Defined Function Group	Audio Output Converter	Audio Input Converter	Pin Complex Widget	Mixer (SumAmp)	Selector (Mux)	Power Widget	Volume Knob	Beep Generator	Vendor Defined Wdiget
Vendor ID	00	R												
Revision ID	02	R												
Subordinate Node Count	04	R	R	R	R									
Function Group Type	05		R	R	R									
Audio Function Group Capabilities	08		o											
Audio Widget Capabilities	09					R	R	R	R	R	R	R	R	R
Sample Size, Rate CAPs	0A		A			A	a							*
Stream Formats	0B		A			A	a							*
Pin Capabilities	0C							R						*
Input Amp Capabilities	0D		A				a	a	a	a				*
Output Amp Capabilities	12		A			a		a	a	a				*
Connection List Length	0E						R	R	R	R	R			*
Supported Power States	0F		R	R	o	o	o	o	o	o	R			*
Processing Capabilities	10					o	o	o		o				*
GPI/O Count	11		o	o	o									
Volume Knob Capabilities	13											R		

Note that the Audio Function Group Capabilities parameter provides a default delay for the entire AFG to be used in lieu of adding specific delays listed for each widget in the Audio Widget Capabilities parameter. This is required if one or more widgets in the AFG opts to not report a correct delay in its Audio Widget Capabilities parameter; if all widgets do report an accurate delay number, the Audio Function Group Capabilities parameter is not required.

Table 113 specifies which verbs and controls are required (R) for each specification-defined node. It also indicates conditional (c) verbs which are required only if the respective optional capability is declared to be available. Another conditional verb (X) is required when the codec supports multiple **SDI** signals. A shaded square in the table indicates that the subject verb is not applicable to the subject node type. Some parameters are marked with an asterisk (*) for the "Vendor_Specific_Audio_Widget" indicating they are not required by the specification since a

vendor specific node may largely define its own verbs. If, however, the vendor specific node implements controls that can be accessed with an existing verb, then using the standard verb is preferable to defining a new one.

Table 113. Required Support for Verbs

Required Verb Support	Get Code	Set Code	Root Node	Audio Function Group	Modem Function Group	Vendor Defined Function Group	Audio Output Converter	Audio Input Converter	Pin Complex Widget	Mixer (SumAmp)	Selector (Mux)	Power Widget	Volume Knob	Beep Generator	Vendor Defined Wdiget
Get Parameter	F00		R	R	R	R	R	R	R	R	R	R	R	R	R
Connection Select	F01	701						c	c		c				*
Get Connection List Entry	F02							R	R	R	R	R	R		*
Processing State	F03	##					c	c	c		c				*
Coefficient Index	D- -	5- -					c	c	c		c				*
Processing Coefficient	C- -	4- -					c	c	c		c				*
Amplifier Gain/Mute	B- -	3- -					c	c	c	c	c			c	*
Stream Format	A- -	2- -					R	R							*
Digital Converter 1	F0D	70D					c	c							*
Digital Converter 2	F0D	70E					c	c							*
Power State	F05	705		R	R	c	c	c	c	c	c	R			c
Channel/Stream ID	F06	706					R	R							*
SDI Select	F04	704					X	X							*
Pin Widget Control	F07	707							R						*
Unsolicited Enable	F08	708					c	c	c	c	c	c	c		*
Pin Sense	F09	709							c						*
EAPD/BTL Enable	F0C	70C							c						*
All GPI Controls	F10 thru F1A	710 thru 71A		c	c										
Beep Generation Control	F0A	70A												R	
Volume Knob Control	F0F	70F											R		
Subsystem ID, Byte 0	F20	720		R	R	R									
Subsystem ID, Byte 1	F20	721		R	R	R									
Subsystem ID, Byte 2	F20	722		R	R	R									

Required Verb Support	Get Code	Set Code	Root Node	Audio Function Group	Modem Function Group	Vendor Defined Function Group	Audio Output Converter	Audio Input Converter	Pin Complex Widget	Mixer (SumAmp)	Selector (Mux)	Power Widget	Volume Knob	Beep Generator	Vendor Defined Wdiget
Subsystem ID, Byte 3	F20	723		R	R	R									
Config Default, Byte 0	F1C	71C							R						
Config Default, Byte 1	F1C	71D							R						
Config Default, Byte 2	F1C	71E							R						
Config Default, Byte 3	F1C	71F							R						
Stripe Control	F24	724					c								
RESET		7FF		R	R	R									

Note that the Connection Select control is not required when the Connection List Length Register value is 1 for this node. In that case, there is no Connection Select control.

7.4 Packages and External Circuits

7.4.1 Standard Audio 48-Pin Codec Packages

The High Definition Audio Specification defines standard packages and pinouts for typical desktop and mobile PC platforms. This specific pinout is not a compliance requirement for all codecs, rather it is intended to maximize physical compatibility between codecs targeted at these specific PC applications.

Table 114 shows the pinout for a standard audio 48-pin codec. The following set of pinout rules is applicable to this table and to the standard 48-pin codec.

1. PORT-A must have output capability.
2. If the codec has headphone capability, it must be available on PORT-A at a minimum.
3. PORT-B and PORT-C, when present, must be microphone re-taskable; i.e., these ports must support VrefOut.
4. On the pinout table, the pin functions which are shaded in blue are required for all standard 48-pin codecs, and these pin function must be on the assigned pin.
5. The non-shaded pin functions are optional; however, if the codec supports this pin function, then it must go on the pin indicated. If the codec does not support this pin function, the pin becomes a vendor option pin.
6. Some pins show two functions, e.g., PORT-*x*/VrefOut-*y*. If the codec supports both pin functions, then the first listed function goes on the indicated pin, and the second function goes on any vendor option pin, vendor's choice. This is the one and only exception to rule #5.

7. The pin function “ExtSpkr” implies a hard wired, codec external amplifier, and the association of the EAPD pin function to this codec port. This requires the Pin Widget for this codec port to have its EAPD capability bit set and to respond to the “EAPD/BTL Enable” verb. There may also be a jack (with detect circuit) on the same port to allow external powered speakers to share a port with a headphone jack (for example). In this case, the “Presence Detect” bit (jack detect) may be used, among other things, in controlling EAPD.
8. The pin function “BTL” indicates balanced output drivers presumably driving hard wired speakers. This implies two output pins for each channel or four output pins for a typical stereo Pin Widget. While the additional outputs and associated amplifiers may be reconfigured from another port or Pin Widget, they must logically appear as a single Pin Widget; i.e., if a stereo Pin Widget was switched to BTL output, it must remain stereo and switch to four output pins rather than two. Similar to the example in rule #7, a headphone jack may be shared with external speakers; in this case, instead of switching an external (EAPD) signal, the Pin Widget would be switched from standard jack (two pin) output to BTL (four pin) output. If a Pin Widget supported BTL exclusively (no sharing), then the “BTL Capable” bit in the Pin Capabilities parameter would be set, and the “Output Capable” bit would be cleared. If the Pin Widget supported headphone jack sharing, then both “BTL Capable” and “Output Capable” bits would be set, and the output mode of the Pin Widget would be controlled by the function driver, after consulting the “Presence Detect” bit.
9. The codec may have “configuration straps” as a vendor option. These allow the codec to configure pins that might have dual purposes (refer to rule #6) to be set up one way or the other. When this is done, the software enumeration view of the pin(s) must be consistent with their electrical and logical operation; i.e., the “straps” must effect not only the operation of the pin(s), but also the parameters defining pin capability. If “configuration straps” are used, they should be applied on pins 33 and 40, and the strap should be a 1-kΩ pull-up resistor to AVdd. A codec-internal circuit should provide a very weak pull-down during link reset (**RST#**); thus, if the external resistor is present, the strap value will be 1, and if the resistor is missing, the value will be 0. Codec configurations which use those pins (33 and 40) as VrefOut pins should define that configuration to be set by strap value = 00. This means the external pull-up, which could be a noise source on an input, would not be applied to a VrefOut.

Table 114. High Definition Audio Codec Pinout

Desktop Configuration	Pin	Mobile Configuration
DVDD_CORE	1	DVDD_CORE
DVSS	2	DVSS
DVDD_IO	3	DVDD_IO
DVSS	4	DVSS
SDO	5	SDO
BITCLK	6	BITCLK
DVSS	7	DVSS
SDI	8	SDI
DVDD_CORE	9	DVDD_CORE
SYNC	10	SYNC
RESET#	11	RESET#
PCBEEP	12	PCBEEP
SENSE_A	13	SENSE_A
PORT-E_L	14	ArrayMic-1
PORT-E_R	15	ArrayMic-2
PORT-F_L	16	ArrayMic-3
PORT-F_R	17	ArrayMic-4
CD-L	18	CD-L
CD-G	19	CD-G
CD-R	20	CD-R
PORT-B_L	21	PORT-B_L
PORT-B_R	22	PORT-B_R
PORT-C_L	23	PORT-C_L
PORT-C_R	24	PORT-C_R
AVDD1	25	AVDD1
AVSS1	26	AVSS1
VREF_FILT	27	VREF_FILT
VrefOut-B_L	28	VrefOut-B_L
VrefOut-C_L	29	VrefOut-C_L
VrefOut-F	30	VrefOut-ArrayMic:3,4
VrefOut-E	31	VrefOut-ArrayMic:1,2
VrefOut-D	32	VrefOut-D/VrefOut-ArrayMic:1,2(2)
VrefOut-G	33	OPTION
SENSE B	34	SENSE B
PORT-D_L	35	PORT-D_L (ExtSpkr <OR> +BTL_L)
PORT-D_R	36	PORT-D_R (ExtSpkr <OR> -BTL_L)
VrefOut-A	37	VrefOut-A/VrefOut-AryMic:3,4(2)
AVDD2	38	AVDD2
PORT-A_L	39	PORT-A_L(ExtSpkr)/+BTL_L
VrefOut-H	40	OPTION
PORT-A_R	41	PORT-A_R(ExtSpkr)/-BTL_L
AVSS3	42	AVSS3
PORT-G_L/VrefOut-D(2)	43	PORT-A/D(+BTL_R)/VrefOut-D(2)
PORT-G_R/VrefOut-A(2)	44	PORT-A/D(-BTL_R)/VrefOut-A(2)
PORT-H_L/VrefOut-C(2)	45	VrefOut-C(2)
PORT-H_R/VrefOut-B(2)	46	VrefOut-B(2)
SPDIF IN/EAPD	47	SPDIF IN/EAPD
SPDIF-OUT	48	SPDIF-OUT

7.4.2 Audio Jack Detection Circuits

The High Definition Audio Specification requires codecs that provide for attachment to more than two pluggable jacks to support the specific means defined in this section of using a single pin to detect the presence of plugs in up to four jacks; up to eight jacks must be detectable with two pins, etc. Each jack must have an isolated switch (normally open), as shown in Figure 84, which closes when a plug is inserted into that jack. A "power of two" parallel resistor network is connected to a single jack detect or "Sense" pin as shown. The codec is required to use the measurable impedance of this network to determine which jacks have plugs inserted and set (or clear) the corresponding "Presence Detect" bit(s) in the "Pin Sense" control (see Section 7.3.3.15) for that Pin Widget or jack. The codec circuitry must appropriately remove switch bounce of up to 250-ms duration. The Pin Widget, if capable of generating unsolicited responses, must deliver *exactly* one such response for each "de-bounced" state change of the "Presence Detect" bit, and the "Presence Detect" bit must be stable and readable at the time such an unsolicited response is delivered.

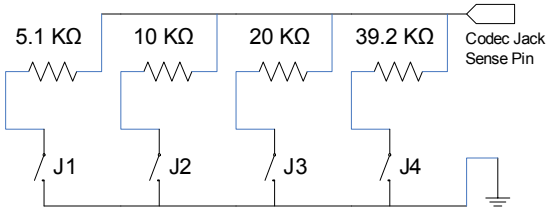

Figure 84. Jack Detect Circuit

In order to ensure physical interoperability between codecs, specific assignment of jacks to Sense Pins and resistor values is required. These are shown in Table 115. Each resistor must be within the tolerance range specified in Table 115..

Table 115. Jack Detect Resistor Assignments

Resistors	Sense Pin A	Sense Pin B
39.2 kΩ ± 1%	Jack-A	Jack-E
20 kΩ ± 1%	Jack-B	Jack-F
10 kΩ ± 1%	Jack-C	Jack-G
5.1 kΩ ± 1%	Jack-D	Jack-H

If a codec supports more than eight jacks, it requires a third Sense Pin. In some cases, S/P-DIF signals may be shared on dual purpose (analog/digital) jacks. When S/P-DIF signals have their own jacks, they may optionally be reported in place of Jack-G (input) and Jack-H (output).

7.5 Codec Power Management Requirements

A Controller Reset (signaled to the codec by the assertion of the **RST#** signal) will not be sufficient for the codec to revert to a primitive initialization state as certain information needs to be retained. For all codecs, the SSID control, if it is writable, must maintain its value through a Reset. For Audio codecs, the Pin Complex Configuration Default must also retain its state. Although modem codecs are not defined as part of this specification, it is expected that they will need to retain Caller ID (CID) data. Codecs will generate their own PWR_GOOD or primary reset condition out of the Aux/Vcc pins.

Codecs (notably the Function Group Widgets) must be able to respond to Power State commands at all power levels, so that they may be brought from a D3 power state to D0.

7.6 Testing Provisions

A unique Link reset exit sequence is defined to place the codec in certain stand alone testing modes, as defined in Figure 85. This test mode is not required, but if a codec defines a test mode, it must use the defined sequence.

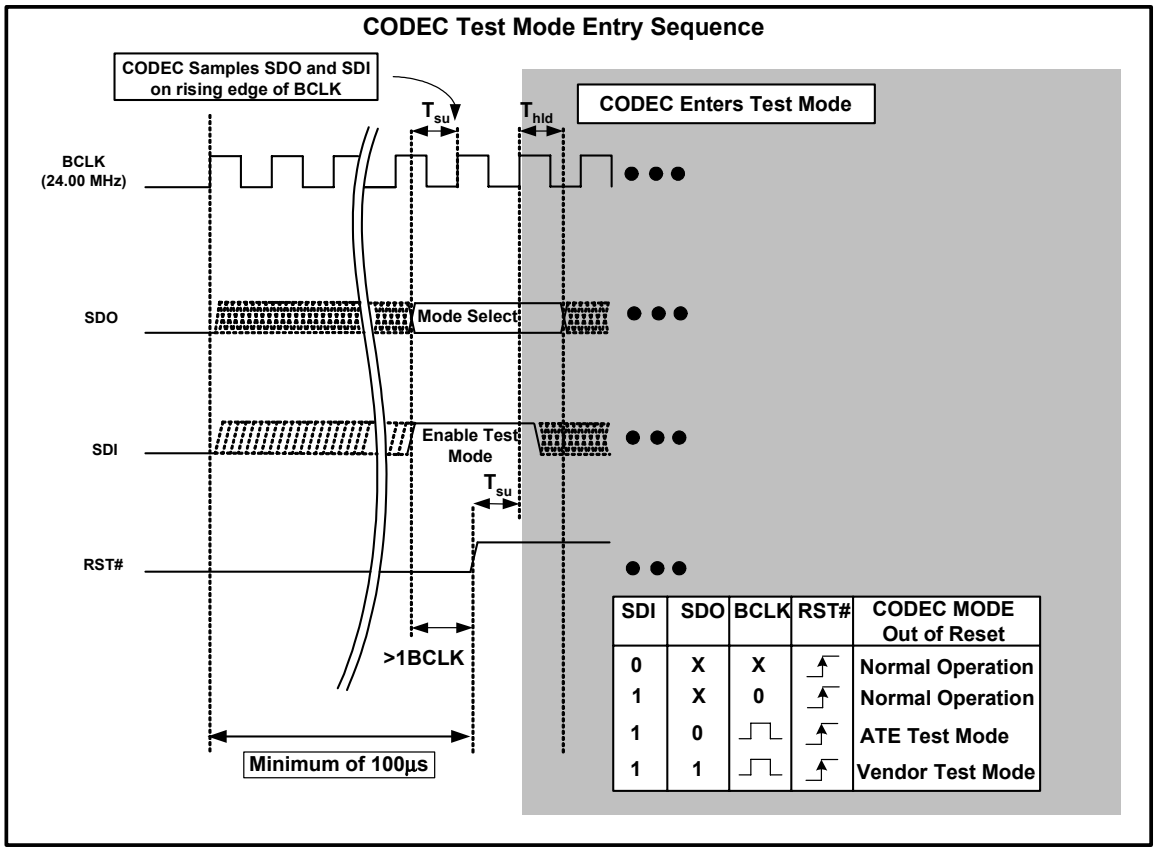

Figure 85. Test Modes