CHAPTER 1 Introduction

This User’s Guide describes PA, provided by Texas Instruments, Inc. (TI). The PA system components include hardware, firmware, software, and documentation that are used to implement an Audio/Video (A/V) receiver development system using an advanced Texas Instruments Digital Signal Processor.

PA can be used in conjunction with development hardware from Spectrum Digital .

PA currently includes , Dolby Digital Plus$^{\text{TM}}$, Dolby Pro Logic IIz$^{\text{TM}}$, , Dolby TrueHD$^{\text{TM}}$Digital Theater System’s DTS Digital Surround$^{\text{TM}}$, DTS-ES Extended Surround$^{\text{TM}}$, DTS Neo:6$^{\text{TM}}$, DTS 96/24$^{\text{TM}}$, and MPEG-II AAC$^{\text{TM}}$, THX Ultra2$^{\text{TM}}$

Note: ***Licensing***

Supply of this Implementation of Dolby Technology does not convey a license nor imply a right under any patent, or any other industrial or Intellectual Property Right of Dolby Laboratories, to use this Implementation in any finished end-user or ready-to-use final product. Companies planning to use this Implementation in products must obtain a license from Dolby Laboratories Licensing Corporation before designing such products

A similar caveat applies to the licensing of DTS, AAC, and MP3 technology.

DA8XEVM firmware are built upon the Performance Audio Framework (PA/F) which is designed to facilitate customer implementation of custom software, firmware, and hardware. This framework is described next.

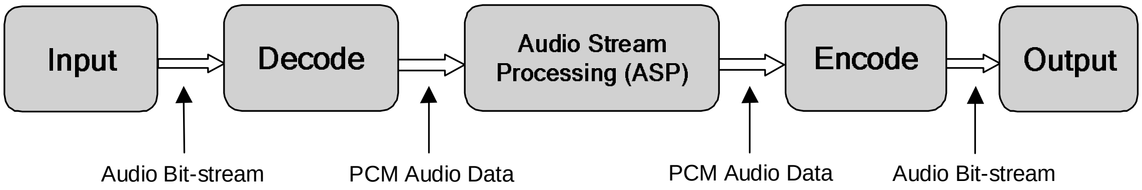
1.1 Performance Audio Framework (PA/F) Overview

The Performance Audio Framework (PA/F) contains all the *components* needed to form a complete digital audio processing solution. The PA Framework facilitates software, firmware, and hardware development at all stages—from prototyping through production systems.

These components can be grouped in a variety of ways. The most fundamental assembly of components makes up a "processing chain" or *Performance Audio Stream* as shown in *Figure 1-1 (Performance Audio Stream)*. Performance Audio is designed to produce symmetric processing systems, with decoding, Audio Stream Processing (ASP), and encoding. These functions, along with input and output functionality, together form the *major components* of the software framework as shown in *Figure 1-1*.

The PA Framework provides control/status communication with these components through a messaging protocol called the *Performance Audio Messaging (PA/M)*, as described in *Section 1.1.4 (PA/F System Communications—Performance Audio Messaging (PA/M))*.

FIGURE 1-1 **Performance Audio Stream**



1.1.1 Firmware Components and Terminology

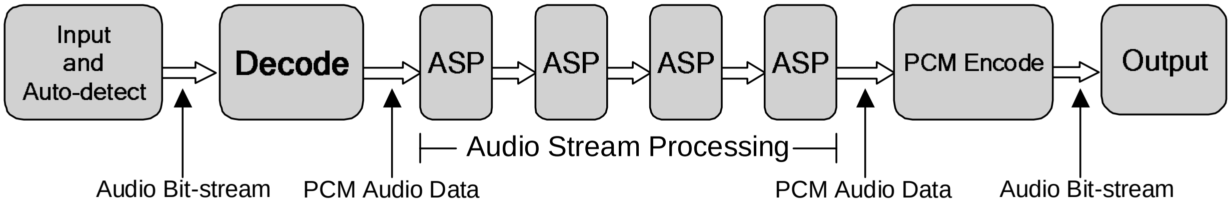
The terminology of these components may be confusing to people who have previously worked with decode-only systems. The following describes the components used in Figure 1-1, and referred to throughout this manual.

Decode Component: This term will be familiar to most people. This component is responsible for decoding the incoming audio stream which could be encoded as PCM, Dolby Digital, or DTS.

Audio Stream Processing (ASP) Component: This term is probably unfamiliar. The ASP Component corresponds roughly to what might be called "post processing" in other systems.

FIGURE 1-2

In the PA Framework, there can be more than one ASP Component in a *processing chain*. The most common manner in which customer-specific code is inserted into the PA Framework is by creating one or more of these ASP Components. Figure 1-2 (In the PA Framework, there can be more than one ASP Component in a processing chain. The most common manner in which customer-specific code is inserted into the PA Framework is by creating one or more of these ASP Components. is similar to Figure 1-1 except that it shows multiple ASP Components in the Performance Audio Stream.Performance Audio Stream Showing Multiple ASP Components) is similar to Figure 1-1 except that it shows multiple ASP Components in the Performance Audio Stream.**Performance Audio Stream Showing Multiple ASP Components**



Decode and Encode Components: A PA Framework requires Decode and Encode Components, even if there is no “decoding” or “encoding” taking place. In other words, if the incoming bit stream is not encoded, there is no need for a decoding function, but there must still be a Decode Component in the system, a “PCM Decoder”. The same is true of the Encode Component. Usually, it will be a “PCM Encoder”, which really isn’t encoding to a compressed bit-stream format but must still be present.

> Although it may seem unnecessary at first glance to include the Decode and Encode Components when no decoding nor encoding is taking place, there are good reasons for doing things this way. One reason has to do with providing a standardized, extensible architecture. Another reason is that the representation of the data is not the same before decoding and after encoding as compared with during Audio Stream Processing. This is explained in the next paragraph.

Audio Bit Stream and PCM Audio Data: The fairly-generic terms, “Audio Bit-Stream” and “PCM Audio Data”, are used for the data flowing between the components shown in Figure 1-1 and Figure 1-2.[1] A “PCM Audio Data” stream is always an unencoded audio sample stream, but an “Audio Bit-Stream” may or may not carry audio data that is encoded or compressed.

Don’t jump to the conclusion that always having the Decode and Encode components will introduce unnecessary overhead—there are functions performed by these blocks that are necessary parts of the system even if no actual decoding or encoding is taking place. For example, a PCM Decoder will still provide control/status to the rest of the system for things like incoming program format, channel configuration, *etc*. Also, a PCM Encoder can provide things like pre-emphasis, volume control, output channel configuration, *etc*.

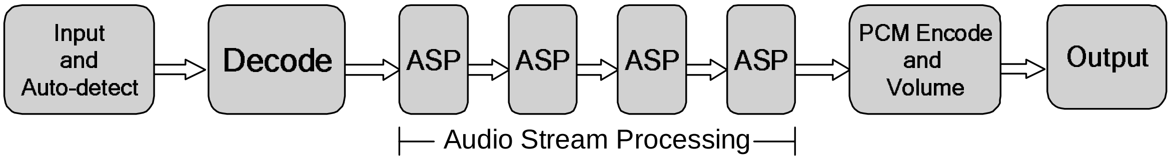
[1] Note that the “PCM Audio Data” are processed using block-based processing rather than sample-based processing. This is an important concept to keep in mind.

1.1.2 PA/F Components in an A/V Receiver

Figure 1-3 (Performance Audio A/V Receiver) shows PA/F components in a typical configuration for an A/V Receiver application.

The Input Component contains an auto-detect capability that can determine what type of encoding, if any, is present in the input bit stream. This information is used to choose a Decode Component that can do the decoding job for the detected compression format. Various ASP Components can be enabled to perform functions like Dolby Pro Logic IIx, Dolby Digital EX, Bass Management, and Speaker Location Delay.[2] The PCM Encode Component also handles the job of volume control.

FIGURE 1-3 **Performance Audio A/V Receiver**



The PA Framework is designed for efficient run-time operation. All components are loaded during system initialization and are available for immediate use during operation. This, for example, allows fast switching between decoders when the incoming program format changes.

The PA Framework contains sophisticated controls for *channel configuration*. The channel configuration describes how the audio data is organized into channels. Many facets of the listening experience can be controlled via the channel configuration, such as the number of channels, the content of those channels, *etc*.

1.1.3 PA/F Extensibility with Standard Component Interfaces

As shown in Figure 1-4 (PA/F Major Firmware Components Use Algorithm and I/O Standards), the major components of PA/F are designed as modules with standardized interfaces. These standardized interfaces are based on TI standard interfaces, that are well-defined and documented. See Section 1.5 (Related Documents from Texas Instruments) for a list of some of these documents.

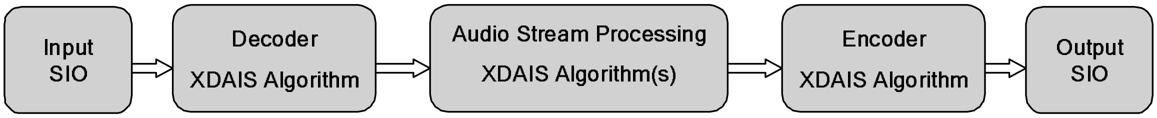
The Decode, Audio Stream Processing, and Encode Components are based on the TI XDAIS Algorithm Standard.[3] These components are realized using Decode Algorithms, Audio Stream Processing Algorithms, and Encode Algorithms. The Input and Output Components are based on the TI Streaming I/O (SIO) device driver standard.

[2] Custom ASP Components can be added using the PA SDK, as documented in the PA SDK User's Guide.

[3] XDAIS is the e**X**press **D**SP **A**lgorithm **I**nterface **S**tandard.

This modular, standardized, extensible architecture provides a uniquely powerful and flexible framework whose primary goal is to provide a high level of processing “intelligence”, but with that intelligence embedded in the modular components, not in the framework itself.

FIGURE 1-4 **PA/F Major Firmware Components Use Algorithm and I/O Standards**



1.1.4 PA/F System Communications—Performance Audio Messaging (PA/M)

The PA Framework provides system communications through a messaging protocol called *Performance Audio Messaging (PA/M).* This messaging mechanism is a vital feature of the framework, providing considerable power and flexibility.

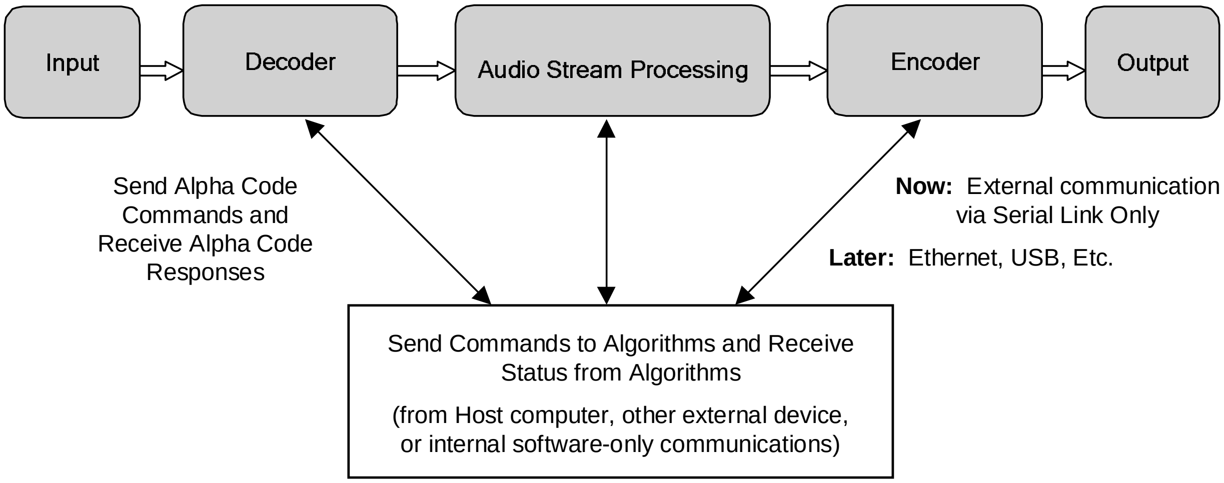
PA/M Alpha Code: The messages themselves are written in something called *alpha code.*[4] Alpha code can be thought of as a simple symbolic command language. One use of this messaging mechanism is to send commands directly to the algorithms that implement the Encoders, Decoders, and Audio Stream Processing (ASP), as shown in .

Note: ***Alpha Code***

Fundamentally, alpha code consists of bit patterns within binary words. However, in actual use, these bit patterns are assigned symbolic names through use of C preprocessor directives such as #define. Thus, though alpha code is not really a symbolic command language, it may be considered as such from a “user”point-of-view.

[4] The term “alpha” here is an adjective meaning *alphabetic*, not a noun meaning s*omething that is first.*

FIGURE 1-5 **Performance Audio Messaging (PA/M) for Software Component Control**


Input
Decoder
Audio Stream Processing
Encoder
Output
Send Alpha Code Commands and Receive Alpha Code Responses
Now: External communication via Serial Link Only
Later: Ethernet, USB, Etc.
Send Commands to Algorithms and Receive Status from Algorithms
(from Host computer, other external device, or internal software-only communications)

PA/M is Bidirectional: Performance Audio Messaging is bidirectional. Alpha code may cause responses to be returned. These responses can be used, for example, to read the status of some part of the audio system, perhaps the current parameters associated with an active Decode Algorithm (program channel content, sample rate, *etc*.).

PA/M Uses Master-Slave or Client-Server Relationship: Performance Audio Messaging is not, however, peer-to-peer. Performance Audio Messaging uses a strictly hierarchical structure, with a clear master-slave or client-server relationship.

PA/M Can be Used with Various Communication Media: Various types of physical communication media and protocols may be used to transfer alpha code from one part of the system to another. Typically, a serial link such as RS-232C, SPI, or I^2C is used as the communication medium. Alpha code can be used to communicate not only between a host and the DSP, but from one DSP to another, or all within the same DSP. In this case, memory is used as the communication medium.

PA/M Shortcuts: Another powerful feature of alpha code is that it can be used to define, and invoke, what are called *shortcuts*. These shortcuts can be used, among other things, to implement complicated configuration presets that can be set up using a single alpha code command.

.

1.1.5 PA/F Algorithms

As described in Section 1.1.3 (PA/F Extensibility with Standard Component Interfaces) above, the PA/F Decode, Audio Stream Processing (ASP), and Encode Components are realized as XDAIS Algorithms. PA/F Algorithms extend the standardized XDAIS Algorithm interface in a further standardized but fully-compliant way to provide the required functionality for digital audio processing.[5]

FIGURE 1-6 **PA/F Algorithm—Generic Block Diagram**

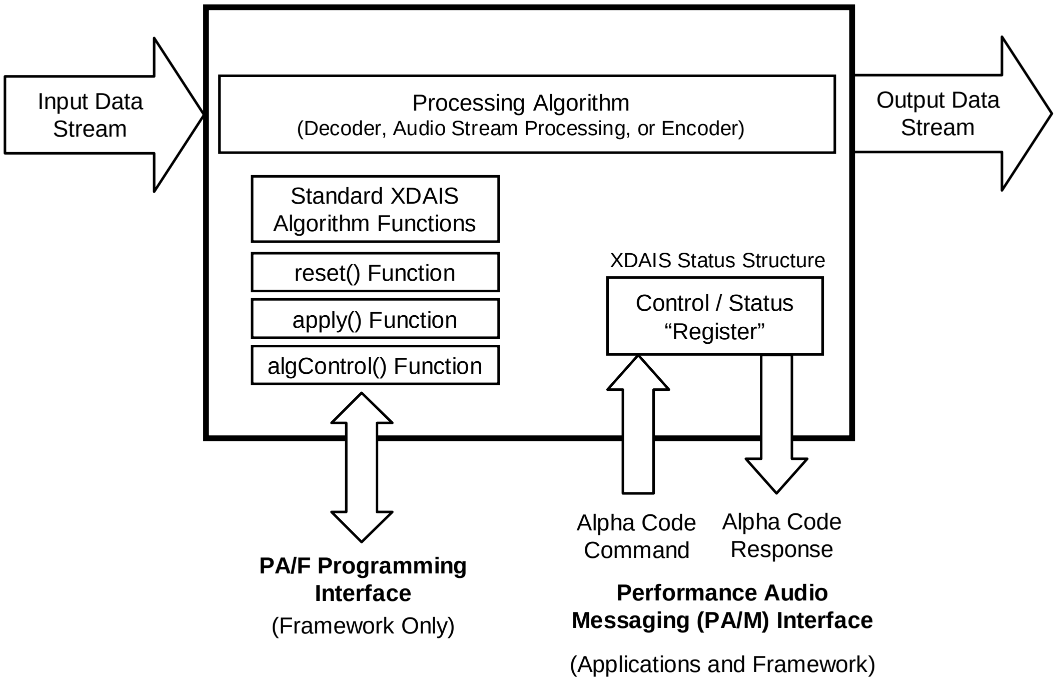


Figure 1-6 (PA/F Algorithm—Generic Block Diagram) shows a "generic" PA/F Algorithm object instance, which could represent an Encode Algorithm, an Audio Stream Processing (ASP) Algorithm, or a Decode Algorithm.

The Performance Audio Messaging (PA/M) Interface forms the Application Interface (API). This is the main method of communications between the application and the algorithms. The term "application", as used here, can mean an application program running on a separate host computer or user code running on the same (or a different) DSP inside the (perhaps embedded) audio processing system. Note also that the PA Framework can also communicate with algorithms through the messaging interface.

This API is documented in the *Application Interface* chapter of the PA User's Guide. Standard PA/F Algorithms are documented in the appendices of the PA User's Guide.

[5] By following PA/F guidelines for extending the XDAIS standard, PA/F-compliant Algorithms can be created and added to PA.

1.1.6 PA/F Streaming I/O (SIO) Device Drivers

PA/F Input and Output Components are based on the TI Streaming I/O (SIO) device driver standard, which is part of the DSP/BIOS API. Basically, the SIO standard provides a “generic” interface layer between the code that actually communicates with a hardware device, and the DSP/BIOS tasks that need to input or output the data.

By using the SIO interface standard, the PA Framework eases the chore of adding new device I/O to the framework. You can read much more about the SIO standard in the following TI documents:

- TMS320C6000 DSP/BIOS User’s Guide
- TMS320C6000 DSP/BIOS Reference Guide.

1.2 PA/F Feature Sets

Feature set refers to features available in the Performance audio stream. It takes into account the decoders, ASPs and encoders available and also the type of IROM patches. In addition it also takes into consideration the method of control communication. PA/F provides for several feature sets. The feature sets supported can be referred to in the Release Notes supplied with the PA/F release.

Additionally, the ASP chain for each Feature Set is documented in the Release notes supplied with the software.

1.3 PA/F Topologies

Topology refers to the configuration of the Performance Audio Stream. The topology takes into account the number of inputs and outputs that are simultaneously available, and also the possibility of multiple independent streams. The name given to each topology usually (but not always) reflects the approximate shape of the letter formed by a pictorial representation, such as a block diagram, of the Performance Audio Stream.

PA/F provides for several topologies, each of which is described below.

1.3.1 I-Topology

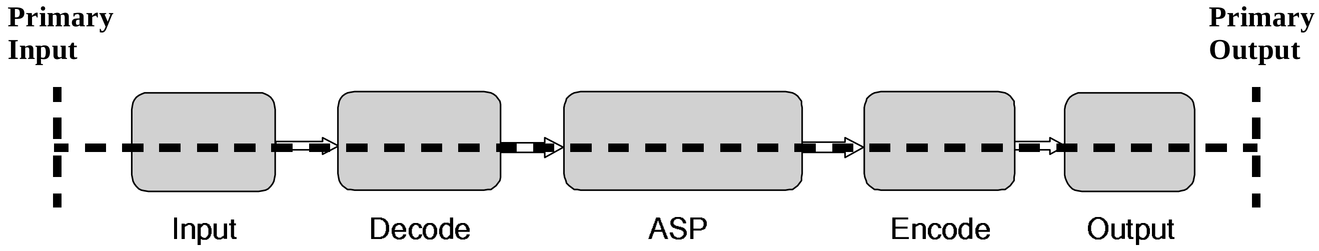
The topology shown in Figure 1-1 and Figure 1-2 is referred to as the I-Topology (uppercase letter “eye”) and is the default for PA. Figure 1-1 is reproduced in Figure 1-7 (I- Performance Audio Stream Configured in the I-Topology), with the letter I superimposed on the block diagram. This clearly shows why this topology is termed the I-Topology.

As can be seen from these figures, the I-Topology provides:

- one *Primary Input*, and

- one *Primary Output*.

FIGURE 1-7 **I- Performance Audio Stream Configured in the I-Topology**



1.3.2 Y-Topology

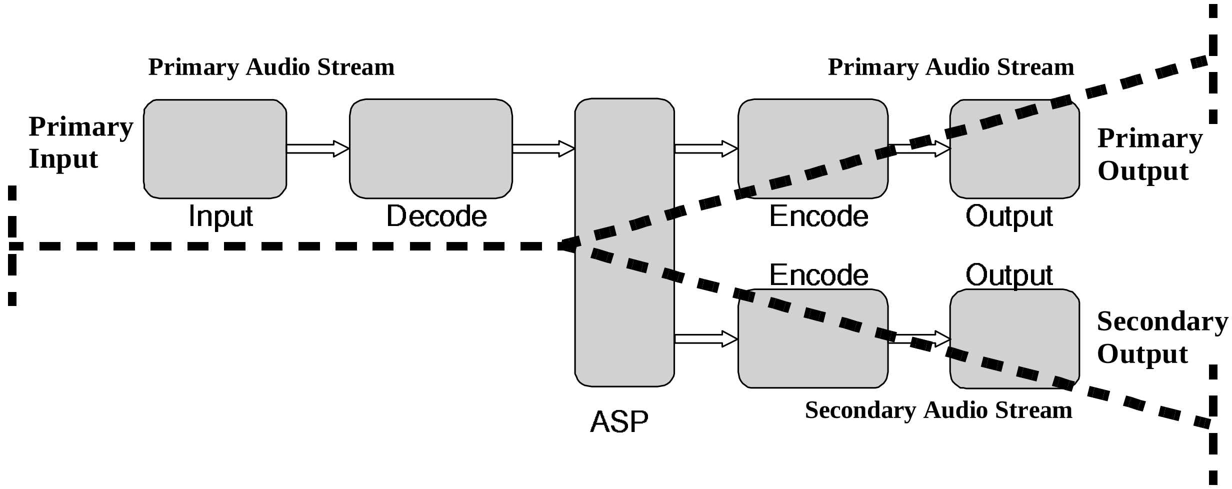
The topology shown in Figure 1-8 (Y- Performance Audio Stream Configured in the Y-Topology) is referred to as the Y-Topology. Figure 1-8 has the letter Y superimposed on the block diagram so as to more clearly show why this topology is termed the Y-Topology.

As can be seen from this figure, the Y-Topology provides:

- one *Primary Input*
- two simultaneous outputs, a *Primary Output* and a *Secondary Output.* The Primary Output is provided by the Primary Audio Stream, whereas the Secondary Output is provided by the Secondary Audio Stream.

The Y-Topology consists of only one Input Component and one Decode Component. However, it has two separate Encode Componets, as well as two separate Output Components.

FIGURE 1-8 **Y- Performance Audio Stream Configured in the Y-Topology**



1.3.3 Comparison Between Topologies

shows a summary of the differences between the various topologies.

TABLE 1-1 **Comparison Between Y and I Topologies**

	I	Y
Number of Input Components	1	1
Number of Decode Components	1	1
Number of ASP Components	1	1
Number of Encode Components	1	2
Number of Output Components	1	2
For use with TI DA8xx processor. .	Yes	Yes

1.3.4 Relation between Topologies and Feature Sets

The following combinations of topologies and feature sets are supported (note that not all combinations are provided as standard configurations in the deliverables, but documentation is provided to describe how to create different supported combinations):

- PAI12- Feature Set 12 in I topology
- PAY13 - Feature Set 13 in Y topology

1.4 PA/F Summary of Features

This section presents some of the PA/F features in condensed, bullet-item format.

1.4.1 PA/F Components and Extensibility

- All major components are extensible per standardized published Application Programming Interfaces (APIs):[6]
    - XDAIS Algorithms: Decode, Audio Stream Processing, Encode.[7]
    - DSP/BIOS Streaming I/O (SIO) drivers.

- New components can be developed using Code Composer Studio (CCS):
    - Documentation and sample code provided.
    - Supplied libraries and header files will facilitate ROM re-use by new components.
    - Components can control/monitor general purpose I/O pins.

- Components can be installed at:
    - "Compile time," using CCS:
        - Statically, using the DSP/BIOS Configuration Tool.
        - Dynamically, using function calls.
    - "Run time," with a sequence of messages which:
        - Allocate memory, download code/data.
        - Configure component settings, start operation.

- Systems communications messaging protocol enables communication with new components.
- Open, extensible Performance Audio Framework (PA/F):
    - TI XDAIS Algorithm interface standard provides "plug-and-play" of algorithms from different sources.
    - Ability to add functionality through ASP Algorithms and other components.

- PA Framework provides great flexibility and speeds time-to-market
    - Built on TI Express DSP Algorithm Interface Standard, which reduces time-to-market.
    - Architecture facilitates artifact-free audio applications.
    - Extensibility allows differentiation of end-equipment.

[6] Not all of these components are user-extendable, but provide extensibility to framework developers.

[7] XDAIS is the e**X**press **D**SP **A**lgorithm **I**nterface **S**tandard.

- Versatile, flexible communications to allow for highly-configurable systems:
  - Host-to-DSP.[8]
  - Within components contained inside one DSP.
  - Between multiple DSPs.

1.4.2 DTS Lock Feature

As per the guidelines from DTS Inc. regarding using DTS technologies, PAF implements feature to lock and unlock usage of them. If any of the DTS technologies are used in the system, then it requires a unique key to unlock and use them. Without this key, the system will not be able to operate properly. The SDK gets distributed without any key and so SDK users need to obtain a key before they can use DTS technologies. However, a system need not have this key if it doesn't use any of the DTS technologies.

1.4.3 Performance Audio Messaging Features

- Simple "register" Read/Write model treats memory locations like hardware registers.
- Configuration presets are facilitated by customer-defined "shortcuts".
- Multiple message types enable bandwidth-efficient communication.
  of 8/16/32/vector data.
- Host/DSP, intra-DSP, inter-DSP communications.

1.5 Related Documents from Texas Instruments

The following are available from TI's web site at http://dspvillage.ti.com.

Hardware:

TMS320C6713 Floating-Point Digital Signal Processor Data Sheet – SPRS186

TMS320DA610 Floating-Point Digital Signal Processor Data Sheet – SPRS002C

TMS320C6000 CPU and Instruction Set Reference Guide – SPRU189

TMS320C6000 Peripherals Reference Guide – SPRU190

Software:

TMS320C6000 Optimizing C Compiler User's Guide – SPRU187

TMS320C6000 Assembly Language Tools User's Guide – SPRU186

TMS320C6000 Chip Support Library API Reference Guide – SPRU401

[8] Although versatile communications between a host CPU and the DSP are easily supported, the DSP can stand alone, and by no means requires a Host CPU for full framework functionality.

TSM320C6000 Programmer’s Guide – SPRU198

TMS320C6000 DSP/BIOS User’s Guide – SPRU303

TMS320C6000 DSP/BIOS Application Programming Interface (API) Reference Guide – SPRU403

TMS320 DSP Algorithm Standard Rules and Guidelines – SPRU352

TMS320 DSP Algorithm Standard API Reference – SPRU360

The best (most up-to-date) information on the Code Composer Tool is found by bringing up CCS and clicking on the ‘Help’ target.

CHAPTER 2 Application Interface - Registers and Messaging

This chapter, along with the following two chapters, introduces the PA Application Programming Interface (API).

Note: ***Documentation on*** *PA*

As previously mentioned in the Note on page 2-27,

- If the description of PA/F in this Reference Guide is applicable to all topologies, the generic name for the Performance Audio Framework, i.e. PA, will be used.

- If the description of PA/F in this Reference Guide is applicable to a particular topology, the specific name, i.e. PAI or PAY will be used.

The above should be kept in mind while reading the rest of this Reference Guide.

The API topics covered in this and the following chapters primarily deal with the *input*, *decode*, *audio stream processing*, *encode* and *output* components. The API topics are divided amongst these chapters as follows:

- Chapter 2 (Application Interface - Registers and Messaging)
- Chapter 3 (Application Interface - Audio Input and Decode)
- Chapter 4 (Application Interface - Audio Encode/Output)
- Chapter 5 (Application Interface - Audio Stream Processing and Volume Control)

In addition:

- Chapter 2 deals with the topics of *Control and Status Registers*, *alpha code*, and *communication* with PA using `calfa` in more detail.

2.1 Register Architecture

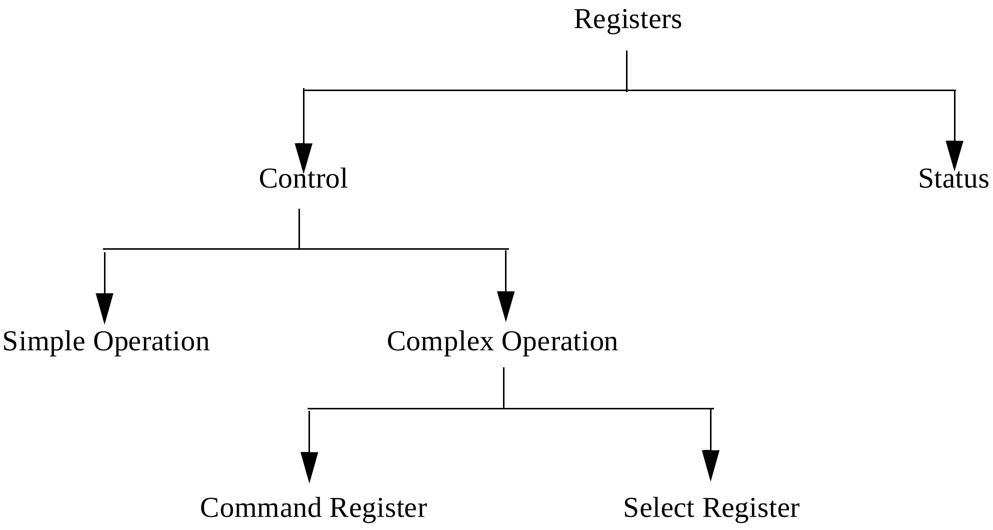
The mode of operation of PA is determined by the values present in a collection of *registers*. Many of the registers discussed below are the Control/Status registers associated with an Algorithm. Other registers are part of the PA Framework itself.

When PA is first powered-up or reset, it enters a default operating mode as defined by the default values of these registers. This mode represents a common operating state for an A/V Receiver. The default setting for each register is highlighted in grey in the tables describing these registers throughout this manual.

2.1.1 Control and Status Registers

A simple classification for control and status registers is shown in Figure 2-1 (Control and Status Registers), and is explained below:

FIGURE 2-1 **Control and Status Registers**



The registers of PA are classified as *control registers* and *status registers*. Generally speaking, control registers *establish* the operating condition of PA, and status registers *report* the operating mode of PA.

- **Control Registers:** Control registers are read-write. That is, both read operations (to determine default or current settings) and write operations (to specify new settings) can be used with control registers.
- **Status Registers:** Read-only registers. Write operations should not be used to change the contents of status registers as undefined behavior may result. Significant exceptions to this general rule must be noted.

- **Select Registers:** Select registers sometimes operate as control registers and other times act as status registers. The type of register they choose to be is based upon what, if any, automatic features (such as the selection of bass management output configuration based upon the speaker control settings) are in use.
- **Command Registers:** Command registers are a special kind of control register that when written to will perform and action and then reset the register to its original value.

Alpha Code. Alpha code is the mechanism by which control of, status extraction from, and actions by PA are effected. The alpha code used to command these registers are classified as *read, write*, and *execution* commands. Collectively, these read, write, and execution commands are known as *alpha code*. The general use of these commands with control and status registers is given in Table 2-1 (General Use of Commands with Control and Status Registers). It is important to keep in mind that the table only reflects the general behavior, and exceptions exist as explained below.

TABLE 2-1 **General Use of Commands with Control and Status Registers**

Commands	Control Register	Control Register Acting as a Status Register	Status Register
Read	Reads value in control register.	Reads values in control register.	Reads values in status register.
Write	Directly alters value in control register.	Generally not used with control register acting as a status register.	Generally not used with status register.
Execution	Performs more complicated operations than simply reading or writing a register. These operations include conditional, state-based, and time-based actions	Generally not used with control register acting as a status register.	Generally not used with status register.

- **Read Commands:** The values present in both control and status registers can be determined using *read commands*. Read commands are typically used to determine the values of control and status register values that result from internal operation. This includes information from the input, as well as information about the output being generated by the system.
- **Write Commands:** The values present in control registers can be altered directly using *write commands*. Write commands will typically cause a change in the operating mode of PA.
- **Execution Commands:** *Execution commands* perform operations that are more complicated than simply reading or writing a register. These operations include conditional, state-based, and time-based actions.

Note: ***Read/Write Restriction Enforcement***

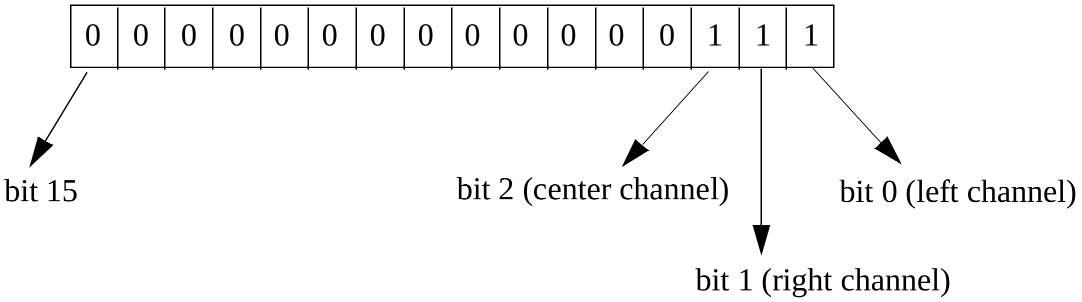
In general, the Performance Audio Framework does not strictly enforce restrictions on the use of various forms of alpha code symbols with various registers. It is the responsibility of the host processor to adhere to use of a limited set of alpha code symbols so that, for example, no write commands to read-only registers are performed.

2.1.2 Bit-mapped Registers

Some of the control and status registers are *bit-mapped registers*. In normal registers, the value of the register is interpreted as a whole. On the other hand, in bit-mapped registers each bit or bit-field in the register represents one setting.

For example, the Listening Format Status Register is a bit-mapped register that reports which output channels are active by setting certain bits within that register. A possible response from this register could be that its contents are the value 0x0007. As shown in Figure 2-2 (Listening Format Status Register Containing the Value 0x0007), this value represents the logical-or combination of bit 0 value 0x0001 which represents the left channel, bit 1 value 0x0002 which represents the right channel, and bit 2 value 0x0004 which represents the center channel. The value 0x0007 thus indicates simultaneously that the left, right, and center output channels are active.

FIGURE 2-2 **Listening Format Status Register Containing the Value 0x0007**



2.1.3 Symbol Definitions

Symbols shown in the text and tables in this chapter are *alpha code symbols*. These are textual descriptions that can be used in place of hexadecimal alpha codes. Alpha code symbols are defined in various C-style *alpha code symbol file*s. The *master alpha code symbol file* for PAI, Feature Set 12, on DA8XEVM with a DA8x device, i12_a.h, for example, encapsulates all needed alpha code symbol files by including all those needed by the product .[9]

2.1.3.1 Symbol Conventions

Almost all alpha code symbols conform to the following conventions:

1. Symbols of the form *`operationXXXRegisterName`*... describe commands for a control or status register called *`RegisterName`* that is part of the beta unit indicated by the symbol *`XXX`*.
2. The *`operation`* is one of `read`, `write`, `wrote`, or `exec`:
   - `read` commands are used to get the contents of control or status registers.
   - `write` commands are used to set the contents of control registers. The values returned by `read` are oftentimes those write commands that are used with those same control registers.
   - `wrote` commands are used to represent the values returned by `read` commands used with status registers.
   - `exec` commands are used to perform more complicated operations than simply reading or writing a register. The return values of exec commands are typically exec commands.

For example, alpha code `readDECSampleRate` specifies a *read* operation of the *Sample Rate* Status Register associated with the *Decode* Algorithm.

Figure 2-3 (Alpha Code Symbol Conventions) shows this common alpha code symbol convention.

If the alpha code symbol name has the letter N at the end, the value specified should be a decimal integer value. However, any C-language expression using C language syntax can be used. For example:

- `writeVOLControlMasterN(-40)` to set the Master Volume Control Register to -20 dB, where -40 is a decimal integer value.

  See Section 5.7.3 (Volume Control Use) for further explanation of `writeVOLControlMasterN( )`.
- `writeVOLOffsetMasterN(0x7fff)` to disable the prevention of mistracking by writing `0x7fff` to the Master Volume Offset Control Register. In this example, `7fff` is a hexadecimal value as indicated by “`0x`” preceding it.

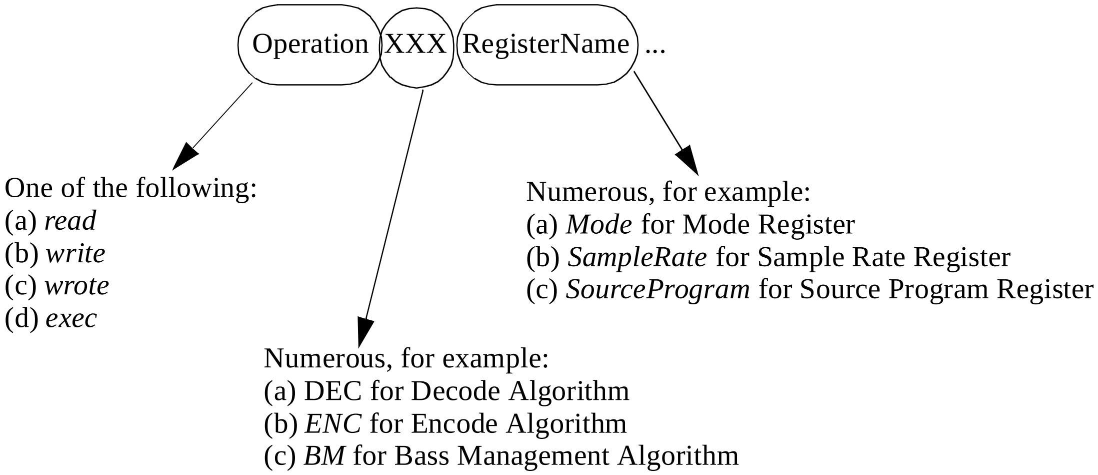
  Note that alpha code `writeVOLOffsetMasterN(7fff)`, where the value `7fff` is specified *without* “`0x`” preceding it, will result in an incorrect value written to the Master Volume Offset Control Register. This can be verified by sending alpha code `readVOLOffsetMaster`, and noting the response.

  See Section 5.7.3 for further explanation of `writeVOLOffsetMasterN( )`.
- `writeBMLFEVolumeN(10)` to set the BM LFE volume level to 5 dB (volume is specified in 1/2 dB increments).

  See Appendix H (Bass Management Algorithm (BM2)) for details about alpha code `writeBMLFEVolumeN( )` and the Bass Management Algorithm.

[9] The special symbol file is used for inverse symbol translation. This file should not be used for other purposes.

FIGURE 2-3 **Alpha Code Symbol Conventions**


Operation
XXX
RegisterName
...
One of the following:
(a) *read*
(b) *write*
(c) *wrote*
(d) *exec*
Numerous, for example:
(a) *Mode* for Mode Register
(b) *SampleRate* for Sample Rate Register
(c) *SourceProgram* for Source Program Register
Numerous, for example:
(a) DEC for Decode Algorithm
(b) *ENC* for Encode Algorithm
(c) *BM* for Bass Management Algorithm

For example, alpha code `readDECSampleRat`e specifies a *read* operation of the *Sample Rate* Register associated with the *Decode* Algorithm.

2.1.4 Register Sets

Register sets are used primarily to effect control of multiple audio streams. Each stream has its own set of registers. This is described below:

PAI : As shown in Figure 2-4 (Single Audio Stream for PAI), PAI provides only a single audio stream, and so the concepts of register sets are not of interest here.

PAY/PAD: As shown in Figure 2-5 (Primary and Secondary Audio Streams for PAY) and Figure 2-6 (Primary and Secondary Audio Streams for PAD), PAY and PAD provide a primary audio stream used for most processing and a secondary audio stream used for generation of secondary output. There are different register sets to control each audio stream, along with a register set that is not associated with either audio stream.

PAH : As shown in Figure 2-7 (Audio Streams 1 and 2 for PAH), PAH provides two audio streams. There are different register sets to control each audio stream, along with a register set that is not associated with either audio stream.

FIGURE 2-4 **Single Audio Stream for PAI**

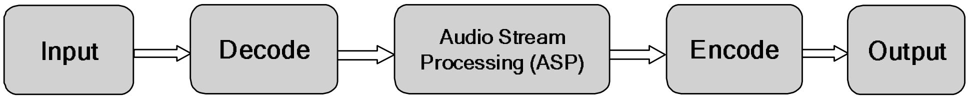


FIGURE 2-5 **Primary and Secondary Audio Streams for PAY**

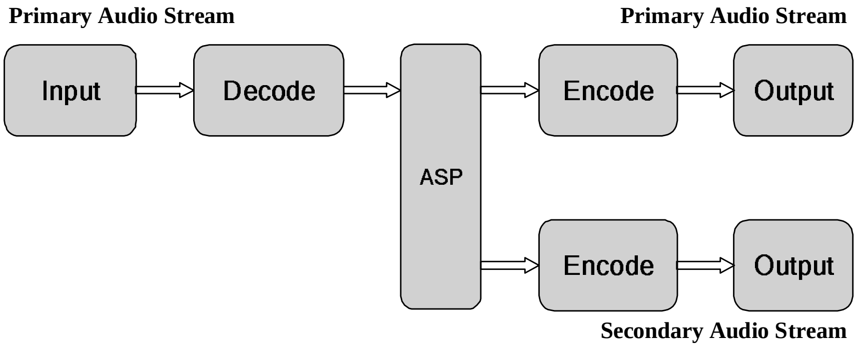


FIGURE 2-6 **Primary and Secondary Audio Streams for PAD**

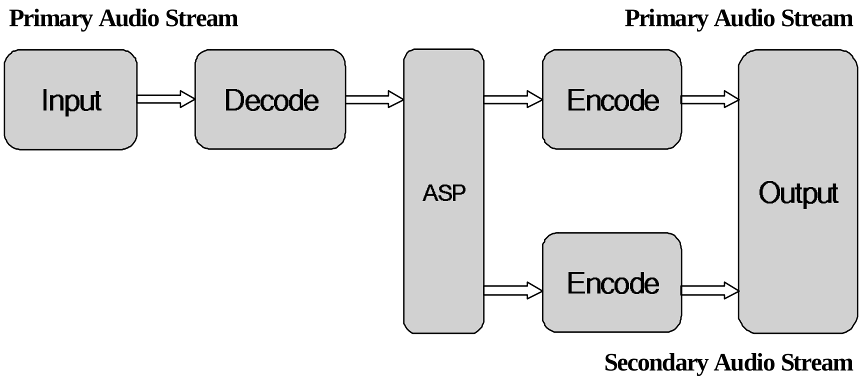


FIGURE 2-7 **Audio Streams 1 and 2 for PAH**

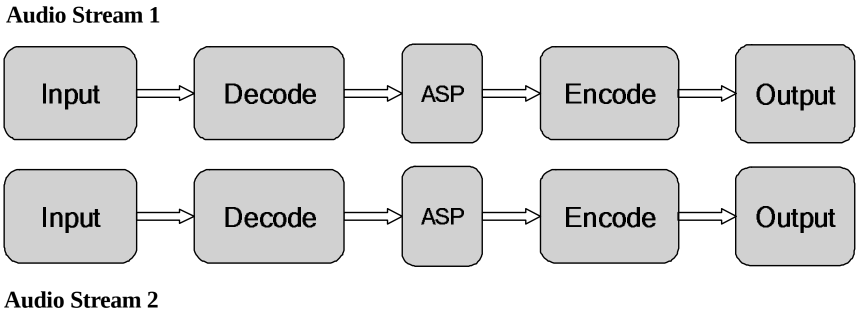
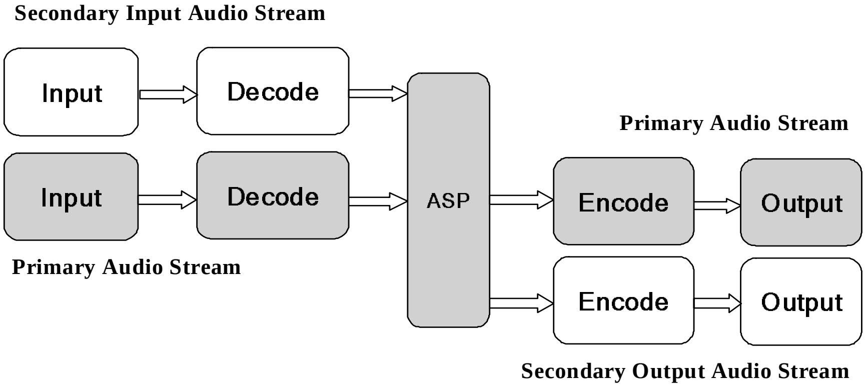


FIGURE 2-8 **Primary and Secondary Audio Streams for PAZ**



Register sets are used primarily to effect control of multiple audio streams. Thus, the registers of PAY and PAH can be divided into three sets:

- The first set of registers consists of those that are not associated with an audio stream. Collectively, these registers are known as the *base register set*. These registers are used as described in Section 2.1.4.1 (Beta Unit Relocation Value) below.
- The second set of registers consists of those that are associated with the primary audio stream. Collectively, these registers are known as the *primary register set*. These registers are used as described in Section 2.1.4.3 (Primary Register Set) below.
- The third set of registers consists of those that are associated with the secondary audio stream. Collectively, these registers are known as the *secondary register set*. These registers are used as described in Section Section 2.1.4.4 (Secondary Register Set) below.

While the registers in the base and primary register sets can, and are, simply used as described in the remainder of this chapter, the registers in the secondary register set are not. One of two, basic techniques are used to access registers in the secondary register set, as described in Section 2.1.4.4.1 (Direct Secondary Register Access) and Section 2.1.4.4.2 (Relocated Secondary Register Access).

2.1.4.1 Beta Unit Relocation Value

The Beta Unit Relocation Value Control Register is used to select the register set to be accessed by subsequent alpha code reads and writes. The value 0 selects the register set associated with the primary audio stream, and the value 1 selects the register set associated with the secondary audio stream.

Practical usage of `calfa` to direct alpha codes to either the Primary or Secondary audio stream is covered in Section 4.4 (Basic Communication Using Calfa).

2.1.4.2 Base Register Set

All user-accessible components in PA that are not part of the primary or secondary register sets are part of the base register set. This includes but is not necessarily limited to the following:

- The Beta Unit Relocation Value Control Register.
- Shortcuts.

2.1.4.3 Primary Register Set

The primary register set includes all of the registers associated with the primary audio stream as described in described in the following chapters, and in the appendices referenced therein.

2.1.4.4 Secondary Register Set

The secondary register set includes all of the registers associated with the secondary audio stream as described in this chapter.

2.1.4.4.1 Direct Secondary Register Access

Secondary registers can be accessed using alpha code that references them directly via their assigned beta unit numbers, that is, using *direct alpha code reads and writes*. Unfortunately, at least for the purposes of discussion here, symbolic alpha code is not supplied with PA that allows direct access to secondary registers.

It is important to understand this access technique, however, as numerical alpha code may be produced that is not completely decompiled. Responses to the relocated alpha code described in the next section are, in fact, direct alpha code.

2.1.4.4.2 Relocated Secondary Register Access

Secondary registers can be accessed using alpha code that references them indirectly via their corresponding beta unit number in the primary register set, that is, using *relocated alpha code reads and writes*. Fortunately, at least for the purposes of discussion here, symbolic alpha code is supplied with PA that allows relocated access to secondary registers.

This symbolic alpha code is, in fact, a combination of symbolic alpha code to set the Beta Unit Relocation Value Control Register of the communications channel and standard symbolic alpha code as used to access the primary registers:

```
alpha 0xcd09,0x0401 // select secondary audio stream
alpha read…,write…,… // standard symbolic alpha code
alpha 0xcd09,0x0400 // select primary audio stream
```

This yields access to the secondary registers using alpha code identical to that used to access the primary registers.

It is quite important to adopt a convention within a single system to either (1) restore the selection of the primary audio stream after any access to the secondary audio stream, as shown above, or (2) precede any audio stream access with the alpha code to select the desired audio stream. The former option is demonstrated above, and the latter below:

```
alpha 0xcd09,0x0400,read…,write…,… // primary audio stream access
alpha 0xcd09,0x0401,read…,write…,… // secondary audio stream access
```

Alternatively, (3) a form can be chosen that is compatible with both options (1) and (2):

```
alpha 0xcd09,0x0401,read…,write…,… // secondary audio stream access
alpha 0xcd09,0x0400,read…,write…,… // primary audio stream access
```

This third form is the standard form used for beta unit relocation. In fact, this form is available automatically using the dual-stream option of `calfa` as described in Section 4.4.1 (Audio Stream-Dependent Communication for ProductNameY/ProductNameH/ProductNameD/ProductNameZ).

2.2 Audio Operations

A description of the *audio operations* provided by PA is given in the following chapters. These audio operations include *input*, *decode*, *audio stream processing*, *encode*, and *output.*The application interface for some audio operations is better presented as a functional view. This functional view is conveniently grouped as *listening mode*, *speaker configuration*, and *volume*, and these are reviewed in the following subsections:

- Section 5.1.1 (Listening Mode)
- Section 5.1.3 (Speaker Configuration)

- Section 5.2 (Basic Audio Stream Processing)

2.2.1 Audio Operation Common Formats

PA provides certain information at several locations along the audio stream. This includes information about:

- the sample rate, and
- the channel configuration,

each of which may be reported at various points along the entire audio stream.

Reporting this information is done utilizing a common format, irrespective of the location along the audio stream at which the information is requested. This common format is explained in this section, and will be referred to in subsequent sections where the *sample rate* and *channel configuration* are specifically discussed.

2.2.1.1 Sample Rate Common Format

The sample rate in all contexts is always reported in a common format as explained below.

All control and status registers that contain sample rate information use the representation shown in Table 2-2 (Sample Rate Symbol Values). As can be seen from the table, in addition to specific sample rate values in units of Hz, the sample rate symbol values also include the description of the sample rate as *Unknown* or *None*. The following notes apply:

- Sample rate *unknown* indicates that there is a sample rate associated with the audio data, but it is not known. Audio data is processed as if the sample rate is known to be 48000 kHz.
- Sample rate *none* indicates that there is not a sample rate associated with the audio data. Any processing that is dependent upon the sample rate is not performed.

TABLE 2-2

Sample Rate Symbol Values

Value	Sample Rate
0	Unknown
1	None
2	32000 Hz
3	44100 Hz
4	48000 Hz
5	88200 Hz
6	96000 Hz
7	192000 Hz
8	64000 Hz
9	128000 Hz
10	176400 Hz
11	8000 Hz
12	11025 Hz
13	12000 Hz
14	16000 Hz
15	22050 Hz
16	24000 Hz

Alpha code symbols are provided as shown in Table 2-3 (Alpha Code for Sample Rate Registers) for all control and status registers that contain sample rate information. The *XXX* in the alpha code symbol names in Table 2-3 is to be replaced appropriately. For example, replacing *XXX* by *DEC* or *ENC* to specify the Decode or Encode Algorithms respectively:

- alpha code `readDECSampleRate` is a request to indicate the input sample rate as determined by the Decode Algorithm.
- alpha code `wroteDECSampleRateUnknown` indicates that the sample rate determined by the Decode Algorithm is unknown, but is treated as 48 kHz.
- alpha code `wroteENCSampleRate44100Hz` indicates that the sample rate at the encoder is 44.1 kHz.

TABLE 2-3

Alpha Code for Sample Rate Registers

Symbol	Description
read*XXX*SampleRate	Determine the sample rate at location *XXX*.
wrote*XXX*SampleRateUnknown	Sample rate is not known but is treated as 48 kHz.
wrote*XXX*SampleRateNone	Sample rate is not known and processing is prevented.

Symbol	Description
wrote*XXX*SampleRate*F*Hz	Sample rate is *F* Hz with *F* determined from Table 2-2 (Sample Rate Symbol Values).
wrote*XXX*SampleRate	Sample rate is something else.

2.2.1.2 Channel Configuration Common Format

The channel configuration is always reported in a common format as explained below.

PA provides information regarding the audio channels present at various points along the audio stream. This information is available via the Channel Configuration Registers.

All control and status registers that contain channel configuration information use the representation shown in Table 2-4 (Alpha Code for Channel Configuration Registers). As can be seen from the table, a vast number of channel configurations is possible including many of the *Phantom* and *Surround* configurations. In addition to the descriptions of specific channel configurations, the table also describes the channel configurations *Unknown* or *None*.

The *XXX* and *Register* in the alpha code symbol names in Table 2-4 are to be replaced appropriately. For example, replacing *XXX* by *DEC* or *ENC* to specify the Decode or Encode Algorithms respectively:

- alpha code `readDECChannelConfigurationProgram` is a request to indicate the channel configuration of the program at the input to the Decode Algorithm, as given by the Decode Channel Configuration Program Register.
- alpha code `readENCChannelConfigurationRequest` returns the value in the Encode Channel Configuration Request Register. This is a control register that sets the form of the output, and controls the number of output channels.
- alpha code `writeENCChannelConfigurationRequestSurround4_1` indicates that the form of the output requested is to provide audio data on the left, center, right, surround, back, and LFE channels.

TABLE 2-4 Alpha Code for Channel Configuration Registers

Symbol	Description
read*XXX*ChannelConfiguration*Register*	Return *XXX* Channel Configuration *Register* value.
write*XXX*ChannelConfiguration*Register*Unknown	The channel configuration is unspecified.
write*XXX*ChannelConfiguration*Register*None	There is no channel configuration, and audio data is carried on no channels.
write*XXX*ChannelConfiguration*Register*Mono	The channel configuration is mono, carried on the Center Channel.
write*XXX*ChannelConfiguration*Register*Stereo	The channel configuration is stereo, carried on the Left and Right Channels.
write*XXX*ChannelConfiguration*Register*StereoLtRt	The channel configuration is surround-encoded stereo, carried on the Left and Right Channels.
write*XXX*ChannelConfiguration*Register*StereoMono	The channel configuration is mono, carried on the Left and Right Channels.

Symbol	Description
write*XXX*ChannelConfiguration*Register*3Stereo	The channel configuration is surround carried on the Left, Center, Right, and LFE or Subwoofer Channels.
write*XXX*ChannelConfiguration*Register*Phantom	The channel configuration is surround carried on the Left, Right, Surround, and LFE or Subwoofer Channels.
write*XXX*ChannelConfiguration*Register*Surround	The channel configuration is surround carried on the Left, Center, Right, Surround, and LFE or Subwoofer Channels.
write*XXX*ChannelConfiguration*Register*None_0	No satellite or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Mono_0	Satellite audio data on Center Channel. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Phantom0_0	Satellite audio data on Left and Right Channels. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Phantom0Stereo_0	Satellite audio data on Left and Right Channels that is not surround-encoded. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Phantom0LtRt_0	Satellite audio data on Left and Right Channels that is surround-encoded. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Phantom0Mono_0	Satellite audio data on Left and Right Channels that is identical. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*PhantomDual0_0	Satellite audio data on Left and Right Channels that is dual mono. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Phantom1_0	Satellite audio data on Left, Right, and one Surround Channel. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Phantom2_0	Satellite audio data on Left, Right, and Surround Channels. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Phantom2Stereo_0	Satellite audio data on Left, Right, and Surround Channels, where the Surround Channels are not surround-encoded. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Phantom2LtRt_0	Satellite audio data on Left, Right, and Surround Channels, where the Surround Channels are surround-encoded. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Phantom2Mono_0	Satellite audio data on Left, Right, and Surround Channels, where the Surround Channels are identical. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Phantom3_0	Satellite audio data on Left, Right, Surround, and one Back Channel. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Phantom4_0	Satellite audio data on Left, Right, Surround, and Back Channels. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Surround0_0	Satellite audio data on Left, Center, and Right Channels. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Surround1_0	Satellite audio data on Left, Center, Right, and one Surround Channel. No LFE or subwoofer audio data.

Symbol	Description
write*XXX*ChannelConfiguration*Register*Surround2_0	Satellite audio data on Left, Center, Right, and Surround Channels. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Surround2Stereo_0	Satellite audio data on Left, Center, Right, and Surround Channels, where the Surround Channels are not surround-encoded. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Surround2LtRt_0	Satellite audio data on Left, Center, Right, and Surround Channels, where the Surround Channels are surround-encoded. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Surround2Mono_0	Satellite audio data on Left, Center, Right, and Surround Channels, where the Surround Channels are identical. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Surround3_0	Satellite audio data on Left, Center, Right, Surround, and one Back Channel. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*Register*Surround4_0	Satellite audio data on Left, Center, Right, Surround, and Back Channels. No LFE or subwoofer audio data.
write*XXX*ChannelConfiguration*RegisterType*_1	Same as above, but with LFE or subwoofer audio data on one channel rather than on no channels.
write*XXX*ChannelConfiguration*Register*HL(Hi16,Lo16)	Channel configuration given as a 32-bit numerical value.

2.2.1.3 Channel Configuration Format for Extended Channels

In addition to the channels mentioned above, the PA implementation also supports the Height Channels pair and Wide Channels pair.

TABLE 2-5 **Alpha Code for Extended Channel Configuration**

Symbol	Description
write*XXX*ChannelConfiguration*RegisterType*LwRw	Same as any channel configuration in Table with the Wide channels
write*XXX*ChannelConfiguration*RegisterType*LwRw	Same as any channel configuration in Table with the Height channels
write*XXX*ChannelConfiguration*RegisterType*LwRwLhrh	Same as any channel configuration in Table with the Height and Wide channels
readDECChannelConfigurationRequestExtchan	Indicate extended channel configuration (downmix) requested of the Decode Algorithm. Is part of DECChannel-ConfigurationRequestRegister.
writeDECChannelConfigurationRequestExtchanNone	Request no extended channels in the channel configuration for the output of the decoder.

Symbol	Description
writeDECChannelConfigurationRequestExtchanLhRh	Request the Height extended channels in the channel configuration for the output of the decoder.
writeDECChannelConfigurationRequestExtchanLwRw	Request the Wide extended channels in the channel configuration for the output of the decoder.
writeDECChannelConfigurationRequestExtchanLhRhLwRw	Request Height and Wide extended channels in the channel configuration for the output of the decoder.

2.3 Serial Control

The Performance Audio Framework provides an interface to control the operations of the DSP software. The PA/F provides three different ways to control the operation; they are UART, SPI and I2C. The ProductName SDK provides a graphical user's interface - SDK Control Panel - using which the user can send Alpha commands to the DSP over the serial link. The PA/F provides user configurability of the serial interface via dcs6_params.c. All the parameters of DCS6 (the driver that handles serial communication) are described in the associated documents - namely - dcs6.html, dcs6_hwsetup.html and dcs6_tp.html. The dcs6 parameters allow the user to configure the communication interface which is suitable for the host controller.

Apart from the serial links mentoned above, on the ARM and DSP devices like DA830, the PA/F provides an interface for the ARM software to control the DSP operation via the shared memory resource between ARM and the DSP.

In all the above cases, the SDK Control Panel provides a convenient way to send alpha commands to DSP for all the communication methods discussed above.

CHAPTER 3 Application Interface - Audio Input and Decode

This chapter, along with the previous chapter, and the next one, introduces the PA Application Programming Interface (API). The API topics covered here primarily deal with the *input* and *decode* components. The API topics are divided amongst these chapters as follows:

Basic and advanced information on the Input and Decode components is provided in this chapter.

3.1 Audio Input

Audio input describes the processing required to present a bit stream to the Decode Algorithm for decoding. Audio input is located at the head of the Performance Audio Stream and is shown shaded in the following figures for each topology:

PAI: Figure 3-1 (Location of the Audio Input Component for PAI)

PAY: Figure 3-2 (Location of the Audio Input Component for PAY)

PAD: Figure 3-3 (Location of the Audio Input Component for PAD)

PAH: Figure 3-4 (Location of the Audio Input Component for PAH)

PAZ: Figure 3-5 (Location of the Audio Input Component for PAZ)

FIGURE 3-1

Location of the Audio Input Component for PAI

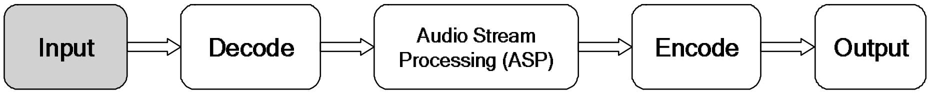


FIGURE 3-2

Location of the Audio Input Component for PAY

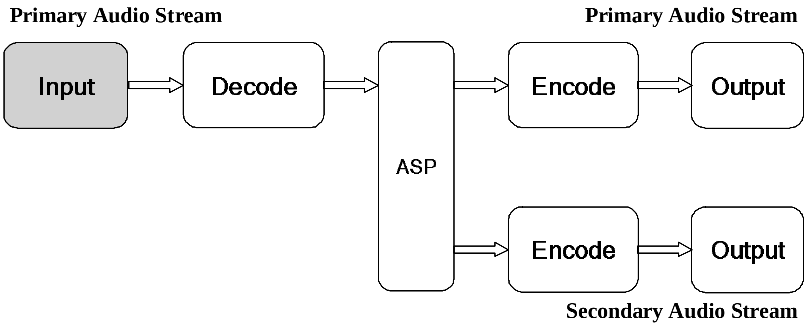


FIGURE 3-3

Location of the Audio Input Component for PAD

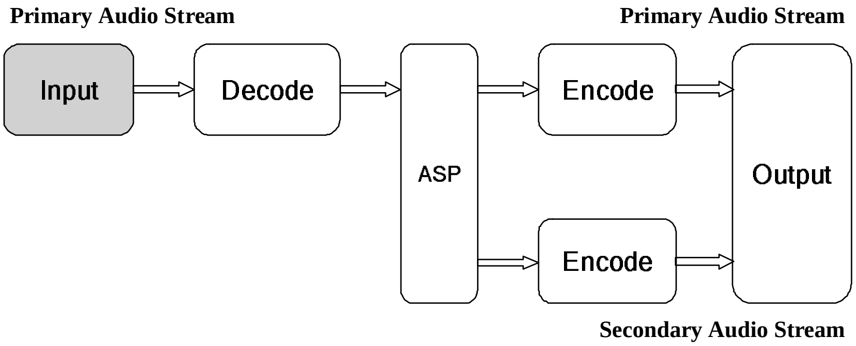


FIGURE 3-4 **Location of the Audio Input Component for PAH**

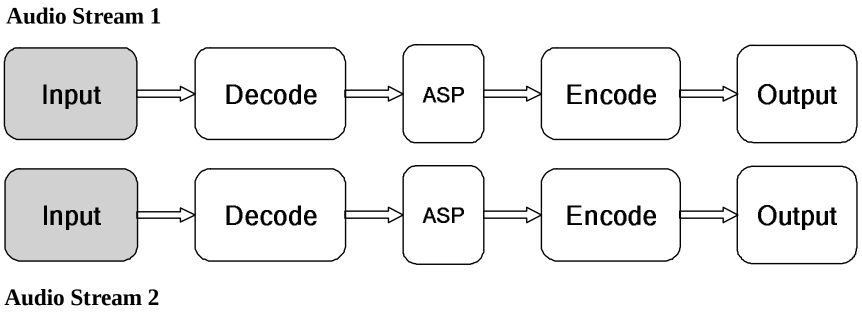
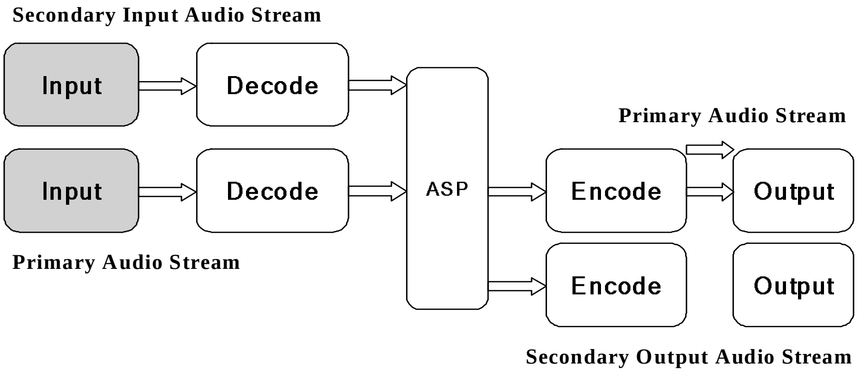


FIGURE 3-5 **Location of the Audio Input Component for PAZ**



3.1.1 Audio Input Shortcuts

PA can handle variety of audio inputs - multichannel/stereo analog, digital, hdmi etc. The inputs are selected using special alpha codes called IOS or Input Output Shortcuts. As these shortcuts are dependent on the hardware on which PA runs, they are discusssed in detail in the separate chapter. Here we will attempt a general introduction to input short cuts. An audio input shortcut is a set of alpha codes which prepares PA for a given type of input. The type of inputs supported on the DA830 hardware are given in Table 3-1.

TABLE 3-1 **Audio Input Shortcuts**

Topology	Alpha Code	Description
PAI/PAY (specified by %)	execPA%InNone	Select no input.
	execPA%InDigital	Select Digital In via optical or coaxial connector
	execPA%InAnalog	Select multi-channel Analog In (48.0 kHz).
	execPA%InAnalogStereo	Select stereo Analog In (48.0 kHz)
	execPA%InSing	Select Signal/Noise Generator
	execPA%InHDMI	Select HDMI input[a]
	execPA%InHDMIStereo	Select stereo HDMI input[b]
	execPA%InRingIO	Select RingIO input from ARM[c]

[a] Available only with HSR4

[b] Available only with HSR4

c. Available only on ARM + DSP devices

IOS Modes are realized using shortcuts to result in the following settings:

- The appropriate Digital, Analog,HDMI etc audio input device is selected. Section 3.1.2 (Audio Input Devices) describes in more detail the various audio input devices that are available with PA
- For digital input, audio precision will be set to that of the input bitstream. For analog input, it is set to the full precision of the ADC devices, 24 bits.
- The sample rate is that of the digital input. For analog input, it is set as indicated to perform Analog-Digital Conversion.The sample rate for Analog-Digital Conversion is specified with the alpha code used to select analog input.
- Pre-emphasis is set to indicate the pre-emphasis state (Y*es/No*) detected from the input audio signal when the input is digital. For analog input, pre-emphasis is set to indicate that the input audio data does not have pre-emphasis present.
- For digital input (including HDMI), the decoder selection is set to Auto. For analog input, it is set to PCM.
- For two-channel analog input, the source type is marked as stereo not-surround encoded (Lo/Ro).
- If a requested IOS Mode is available, the identical execution command is returned.

  For example, sending alpha code `.`
- If a requested IOS Mode is not available, the error value `execPA%InOutError` is returned.

  The error value, `execPA%InOutError`, is used to return error status. It should not be used as an execution command. That is, alpha code `execPA%InOutError` should *not be sent* from the host to the target processor. It is used only to return error status from the target to the host processor.
- If a serious error occurs during processing of the shortcut, the response `0xdead` will be returned. A time-out on the communications link may indicate a fatal error within the system.

3.1.2 Audio Input Devices

The Input Buffer SIO Select Command Register, which is a part of IOS, described in , indicates the audio input device that is currently *selected* for use, or *pending* for use.

Register	Alpha Code	Description
Input Buffer SIO Select[a]	readIBSioSelect	Indicate the input device set selection for this audio stream .
	wroteIBSioSelect*X*	Current input device set where X determines the input to be selected.
	wroteIBSioCommand*X*	Pending input device set selection *X*.

[a] Despite its name, this is a command register and not a select register.

3.1.3 Audio Input Source Modes

PA provides information about the source material presented at the input for decoding and processing. This source material is known as the *program*. Information regarding the source program is provided in Section 3.2.2 (Audio Decode Source Modes).

3.1.4 Audio Input Pre-Emphasis

PA provides information about the pre-emphasis of the input bit stream. Alpha code symbols for determining whether or not the input bitstream indicates that pre-emphasis filtering has been applied is provided in Table 3-2 (Alpha Code for Audio Input Pre-emphasis).

Send alpha code `readIBEmphasisStatus` to determine whether or not the input bitstream indicates that pre-emphasis filtering has been applied. The response `wroteIBEmphasisStatusNo` or `wroteIBEmphasisStatusYes` indicates a `No` or `Yes` respectively.

See Section 8.1.8 (Audio Input Pre-emphasis) for the description of other alpha codes for audio input pre-emphasis.

TABLE 3-2 **Alpha Code for Audio Input Pre-emphasis**

Register	Alpha Code	Description
Emphasis Status	readIBEmphasisStatus	Indicate whether or not the input bitstream indicates that pre-emphasis filtering has been applied or not.
	wroteIBEmphasisStatusNo	Input bit stream did not have pre-emphasis filtering applied.
	wroteIBEmphasisStatusYes	Input bit stream had pre-emphasis filtering applied.

Alpha codes for Audio Input Pre-emphasis are provided and described in Table 3-3 (Alpha Code for Audio Input Lock). As can be seen from Table 3-3, there are several registers for pre-emphasis. An explanation of the alpha codes used with the following registers from this table is covered here:

- Emphasis Data
- Emphasis Override
- Emphasis Status

TABLE 3-3 **Alpha Code for Audio Input Lock**

Register	Alpha Code	Description
Emphasis	readIBEmphasis	Returns the contents of the 8 bit EmphasisData Status register. If the value is EmphasisDataYes then the input stream will be processed with pre-emphasis. If the value is EmphasisDataNo then the input stream will not be processed with pre-emphasis.
Emphasis Data	readIBEmphasisData	Indicate if the input bitstream indicates that pre-emphasis filtering was applied.
	wroteIBEmphasisDataUnknown	Presence of pre-emphasis in input bit stream is not known. This state may be reported during initialization of input processing and prior to application of an input bit stream.
	wroteIBEmphasisDataNo	Input bit stream did not have pre-emphasis filtering applied.
	wroteIBEmphasisDataYes	Input bit stream had pre-emphasis filtering applied.
Emphasis Override	readIBEmphasisOverride	Return indication of whether the bitstream pre-emphasis indicator is used or overridden.
	writeIBEmphasisOverrideDisable	No override - use the pre-emphasis indication given by the bitstream.
	writeIBEmphasisOverrideNo	Override the detected pre-emphasis by indicating that there is NO input pre-emphasis.
	writeIBEmphasisOverrideYes	Override the detected pre-emphasis by indicating that there IS input pre-emphasis.
Emphasis Status	readIBEmphasisStatus	Indicate whether input bitstream is considered to have pre-emphasis filtering applied or not.
	wroteIBEmphasisStatusNo	Input bit stream did not have pre-emphasis filtering applied.
	wroteIBEmphasisStatusYes	Input bit stream had pre-emphasis filtering applied.

The Input Buffer Emphasis Data Register indicates whether or not the input bitstream has pre-emphasis filtering applied to it. This pre-emphasis value is determined from the selected input device. This register is initialized following input device configuration, then periodically updated thereafter. The Input Buffer Emphasis Override Select Register controls whether the detected pre-emphasis of the input bit stream should be used to specify if emphasis processing should be applied, or if it should be turned on or off regardless of what the input bitstream indicates. Whether pre-emphasis filtering should be considered applied or not is indicated in the Input Buffer Emphasis Status Register.

When digital input is selected, the Input Buffer Emphasis Data Select Register will indicate if pre-emphasis status data has been detected in the input bitstream. By default, IOS will set the Input Buffer Emphasis Override Control Register to `wroteIBEmphasis-OverrideDisable`, which causes the Input Buffer Emphasis Status Select Register to be set to reflect the pre-emphasis information detected in the input.

For analog input is selected, the Input Buffer Emphasis Data Select Register indicates that pre-emphasis status data is not present. By default, IOS sets the Input Buffer Emphasis Override Control Register to always indicate that there is no pre-emphasis, which causes the Input Buffer Emphasis Status Select Register to be set to `wroteIBEmphasisStatusNo`.

Following initialization by IOS, the Input Buffer Emphasis Override Control Register can have its setting altered. Not that each time the input is changed from Analog to Digital, this register will have to be reset to the desired setting if it is other than the IOS default.

The value of the Input Buffer Emphasis Status Register controls the on/off switch for the De-emphasis Audio Stream Process (ASP). Automatic De-emphasis and associated registers are described in Appendix C.2.1 (Automatic De-emphasis (Default)).

3.1.5 Audio Input Sample Rate

PA provides information about the sample rate of the input bit stream.

The SIO Sample Rate Status Register indicates the sample rate of the input bit stream. This *audio input sample rate* is determined by the SIO Input Device. Different SIO Input Devices may determine this sample rate in different ways.

- TBD.

For more details on the method used to determine the sample rate by a specific SIO Input Device, see the appendix for that device.

3.1.6 Audio Input Precision

PA provides information regarding the precision of the input signal. Alpha code for Audio Input Precision are provided and described in Table 3-4 (Alpha Code for Audio Input Precision). As can be seen from Table 3-4, there are several registers for precision. An explanation of the alpha codes used with the following registers from this table is covered here:

- Precision Default
- Precision Detect
- Precision Override
- Precision Input

TABLE 3-4 **Alpha Code for Audio Input Precision**

Register	Alpha Code	Description
Precision Default[a]	readIBPrecisionDefault	Return Input Buffer Precision Default Control Register value.
	writeIBPrecisionDefaultOriginal	When precision cannot be determined, operate with IEC 60958 word length default: 16 bits.
	writeIBPrecisionDefaultStandard	When precision cannot be determined, operate with IEC 60958 word length default: 20 bits.
	writeIBPrecisionDefaultFull	When precision cannot be determined, operate with S/PDIF word length: 24 bits.

Register	Alpha Code	Description
Precision Detect	readIBPrecisionDetect	Indicate the precision detected in the input bit stream.
	wroteIBPrecisionDetectTBD	This feature is not yet implemented.
Precision Override	readIBPrecisionOverride	Return Input Buffer Precision Override Control Register value.
	writeIBPrecisionOverrideDetect	No override—use the detected precision as the input precision.
	writeIBPrecisionOverrideDefault	Override the detected precision and use the default precision as the input precision.
	writeIBPrecisionOverride*N*	Override the detected precision and use the indicated precision as the input precision, *N*=16–24.
	writeIBPrecisionOverrideFloat	Override the detected precision and indicate that the input data is floating point. Note this setting should only be used for PCM data.
Precision Input	readIBPrecisionInput	Indicate the precision used to process the input bit stream.
	wroteIBPrecisionInputBitStream	The input bit stream is processed as other than PCM audio data with 16 bit precision.
	wroteIBPrecisionInput*N*	The input bit stream is processed as PCM audio data with precision: *N* bits.

[a] For information on default settings for this control register, see Section 3.1.1 (Audio Input Shortcuts).

PCM audio data carried via S/PDIF or other digital media, as well as that produced via ADC, must be processed variously according to its precision.[10] PA extracts information regarding precision from input bit streams, and this information is used to control the processing applied to that input. Complete information about the manipulation of control and determination of status regarding the precision of the input bit stream is provided in this section.

The SIO Precision Control and Status Registers are used together to achieve the following effects:

- For non-PCM input bit streams, 16-bit precision is always used.
- For PCM input bit streams, precision information present in the bit stream is extracted and utilized as part of the operation of the SIO Input Device. When such information is not available, a local precision default is utilized.[11]
- For locally-generated PCM input, such as ADC, a local precision override is utilized.

The local precision selections are described briefly in Section 3.1.1 (Audio Input Shortcuts).

[10] Precision is the preferred term in the context of Performance Audio. The term *word length* is used in IEC documents, and other terms are used elsewhere.

[11] In the current version of PA, the detection feature is not implemented, and the default value is always used.

Audio Input Precision Default Control

The Input Buffer Precision Default Control Register value is used as the input precision if (1) a valid precision cannot be detected and (2) the precision is not overridden.

The Input Buffer Precision Default Control Register is typically set to Original, Standard, or Full for digital input jacks designed primarily for corresponding input types.

Audio Input Precision Detect Status

The Input Buffer Precision Detect Status Register value indicates the input precision detected in the input bit stream.[12]

For details on the method used to determine the precision by a specific SIO Input Device, see the appendix for that device.

Audio Input Precision Override Control

If the Input Buffer Precision Override Control Register is set to Detect or Default, then the corresponding Input Buffer Precision Detect Status Register or Input Buffer Precision Default Control Register is used as the input precision, respectively. Otherwise, the value of the Input Buffer Precision Override Control Register is used as the input precision.

The Input Buffer Precision Override Control Register is typically set to Detect for input from a bit stream source and to the appropriate precision for input from an analog source.

When the Input Buffer Precision Override Control Register is set to ‘Float’ (using the alpha code writeIBPrecisionOverrideFloat), it indicates that the input data is floating point. This setting should only be used for PCM input. The PCM decoder converts input fixed-point data to floating point based on the precision. If the input precision is set to Float, the PCM decoder does not perform the fixed-to-float operation. The envisioned use of this control is in Multi DSP systems. In a Multi DSP system, it may be required that the DSP receive input data that is floating point. In such scenarios the Float setting for Input Buffer Precision Override Control Register can be used.

Audio Input Precision Input Status

The Input Buffer Precision Input Status Register indicates the input precision that is used.

3.1.7 Audio Input Channel Configurations

PA provides information regarding the audio channels present in the source material. This section augments the information regarding the channel configuration of the program provided in Section 5.2.3 (Audio Decode Channel Configuration).

Decode Channel Configuration Program

The Decode Channel Configuration Program Status Register indicates the channel configuration of the program at the input to the Decode Algorithm. Various parameters are extracted from the source bit-stream to determine which channels are present.

[12] In the current version of PA, the detection feature is not implemented, and this register always indicates `TBD`.

The Channel Configuration Program Status Register can be accessed as four byte-wide sub-registers, corresponding to the satellite, subwoofer, auxiliary, and reserved portions of the channel configuration from least- to most-significant byte, respectively.

Table 4-3 (Alpha Code for Channel Configuration Registers) lists the alpha code for use with the Channel Configuration Registers. In particular, alpha code for use with the Decode Channel Configuration Program Status Register is obtained from those listed in this table by:

- replacing *XXX* by *DEC*, and
- replacing *Register* by *Program*.

As an example, alpha code `writeDECChannelConfigurationProgramUnknown` indicates that the channel configuration is unspecified.

3.1.8 Audio Input Channel Map

The Decode Channel Map From Control Register is used to determine the manner in which the channels of the SIO Input Device are mapped onto the input channels of the Decode Algorithm. It is used in conjunction with the Decode Channel Map To Control Register as described in Section 3.2.6 (Audio Decode Channel Map) in the manner discussed in Section 4.3.1 (Channel Maps).

This mapping is flexible. For each of the channels to be decoded, either no SIO channel or any particular SIO channel can be mapped.

The Decode Channel Map From Control Register consists of 16 8-bit sub-registers, of which only the first 8 are pertinent. While up to 16 sub-registers can be manipulated either singly or in a group, in practice N_d sub-registers are set simultaneously with a single alpha code as shown in Table 3-5 (Alpha Code for Audio Input Channel Map), where N_d is the number of sub-registers to be used by the Decode Algorithm: $N_d = 2$ except for multi-channel PCM input where $N_d = 8$. Each sub-register can take on one of two types of values. A negative value signals that no SIO channel is to be mapped onto this decode-input channel for use by the Decode Algorithm.[13] A non-negative value signals that this SIO channel is to be mapped onto this decode-input channel for use by the Decode Algorithm.

[13] In this case, the corresponding Decode Channel Map To Control Register value is ignored.

TABLE 3-5 **Alpha Code for Audio Input Channel Map**

Register	Alpha Code	Description
Channel Map From	readDECChannelMapFrom	Return Decode Channel Map From Control Register value.
	writeDECChannelMapFrom16(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15)[a]	Select default input channel map for 2-channel input.
	writeDECChannelMapFrom2(...)	Select standard input channel map for *N* channels, *N*=2, 8, or 16. These macros take *N* arguments each.
	writeDECChannelMapFrom8(…)	
	writeDECChannelMapFrom16(…)	
	wroteDECChannelMapFrom,...	Used to indicate the Decode Channel Map From Control Register value.

[a] The default value is as shown for the primary audio stream but otherwise for the secondary audio stream.

Examples best serve to illustrate the operation of the Decode Channel Map From Control Register:

- Consider the default input channel map shown in Table 3-5. This is reproduced below:

  writeDECChannelMapFrom16(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15)

  In fact, however, this is overkill. This is because only the first N_d of the sixteen entries are used for N_d-channel input by the Decode Algorithms. One of the following input channel maps will adequately specify what is required to reproduce such input. In the case of the default, 2-channel input channel map immediately below will suffice.

- Consider the following input channel map used for 2-channel input, as to the PCM Algorithm or Bit-Stream Decode Algorithms:

  writeDECChannelMapFrom2(0,1)

  The effect of this *2-channel input channel map* is shown in Table 3-6. In this case, elements 0–1 are use for the 2-channel input, elements 2–7 are ignored, and elements 8–15 are not pertinent in PA.

- Consider the following alternate input channel map used for 6-channel input, as to the PCM Algorithm:

  writeDECChannelMapFrom8(0,3,1,4,2,5,-1,-1)

  The effect of this *6-channel input channel map* is shown in Table 3-7. In this case, elements 0–5 are used for 6-channel, interleaved input, and elements 6–7 are ignored. As above, elements 8–15 are not pertinent in PA.

- Consider the following alternate input channel map used for 8-channel input, as to the PCM Algorithm:

  writeDECChannelMapFrom8(0,4,1,5,2,6,3,7)

  The effect of this *8-channel input channel map* is shown in Table 3-8. In this case, elements 0–7 are used for 8-channel, interleaved input. As above, elements 8–15 are not pertinent in PA.

The settings that result from use of the alpha code symbols shown above, that is, writeDECChannelMapFromN_d, are appropriate for use with the SIO Input Driver provided with PA. They all require, however, an appropriate setting of the PCM Algorithm to force N-channel decoding, $N <= N_d$.

See also Section 3.2.6 (Audio Decode Channel Map) and Section 4.3.1 (Channel Maps).

TABLE 3-6

2-Channel Input Channel Map

Bit-Stream Channel	Logical Audio Channel
1	Left
2	Rght

TABLE 3-7

6-Channel Input Channel Map

Bit-Stream Channel	Logical Audio Channel
1	Left
2	LSur
3	Cntr
4	Rght
5	RSur
6	Subw

TABLE 3-8

8-Channel Input Channel Map

Bit-Stream Channel	Logical Audio Channel
1	Left
2	LSur
3	Cntr
4	LBak
5	Rght
6	RSur
7	Subw
8	RBak

3.1.9 Audio Input Unknown Timeout

When the Source Select Control Register is configured for Automatic Source Select (e.g., via `writeDECSourceSelectAuto`), the Audio Input passes through a process known as auto-detection. This process determines the input source type and sets the Decode Source Program Status (obtained via `readDECSourceProgram`) appropriately. See Section 5.2.2 (Audio Decode Source Modes) for details on the Decode Source Program Status Register.

If a known bitstream is not detected after a specified number of stereo samples, then the autodetection process times-out. When this occurs, the Decode Source Program Status Register is changed from Unknown to PCM. In other words, when the first non-zero sample is detected, a counter starts counting stereo samples, and if that counter reaches the number specified by the Unknown Timeout Control Register, the autodetection process declares that the input is PCM. If, prior to the counter reaching the Unknown Timeout Control Register value, autodetect determines that the bitstream is not PCM, it sets the Decode Source Program Status Register to indicate the bitstream type, and the count is abandoned.

Values of the Unknown Timeout Control Register less than the current bitstream input frame length cause unreliable bitstream detection. This is because the bitstream identification is typically located at the beginning of each frame; if the first few samples at the beginning of the frame are missed (due to randomness of switching to the new bitstream, for example), autodetect will have to examine the bitstream until it sees the start of the next frame and the identification in it, if it is to correctly classify the input.

Therefore, small values of Unknown Timeout are neither feasible nor recommended if bitstream decoding is required. Such values result in almost-immediate entry into the PCM state, and resulting play of the bitstream as PCM (which is undesirable). Even though in the PCM state, the audio input continues to be monitored for bitstream information, and eventually the correct bitstream detection will occur. When this happens, PCM decoding will stop and the appropriate bitstream decoder will be invoked, if present in the build; if the bitstream decoder is not present in the build, the output will simply be muted.

This register is a 32-bit unsigned integer in units of stereo samples with a maximum value of 0x7FFFFFFF.

Increasing this register from the default value increases the number of initial PCM samples that are discarded when a PCM bitstream is input. Decreasing this register from the default value increases the likelihood that non-PCM-bitstream input will be mistakenly declared to be PCM, producing undesirable audio output. The default value provides a reasonable compromise.

Alpha code for use with the Unknown Timeout Control Register is given in Table 3-9 (Alpha Code for Audio Input Unknown Timeout).

TABLE 3-9 **Alpha Code for Audio Input Unknown Timeout**

Register	Alpha Code	Description
Unknown Timeout	readIBUnknownTimeout	Return Unknown Timeout Control Register value as described in Section 3.1.9 (Audio Input Unknown Timeout).
	writeIBUnknownTimeoutN(NN)	Set the Unknown Timeout Control Register. NN=0 is not supported.
	writeIBUnknownTimeoutN(2048)	Select default Unknown Timeout.

3.1.10 Audio Input Zero Run Control

Many DVD players transition between PCM and bitstream output without a clock change or otherwise signalling such a transition. If this transition is not detected then a portion of the bitstream will be played as PCM and result in unacceptable noise to the listener. Fortunately such transitions can often be characterized by a contiguous stream of zero samples; which in PA terminology is a *zero run*. Therefore when, as a result of Automatic Source Select, the Source Program Status Register indicates PCM mode, the input is scanned for the presence of a zero run. Once in the zero run state the system continues to decode but the input is replaced by zeros. This is important since the purpose is avoid immediate decoding of non-zero samples which follow the zero run.

While in the PCM state, there is one way to enter the zero run state and three ways to exit. It is entered only when the number of zeros equals that specified by the Zero Run Trigger Control Register. Exit possibility one is that the number of zeros equals that specified by the Zero Run Restart Control Register. When this happens decoding stops, the Source Program Status Register indicates Unknown, and the system re-enters autodetection. Exit possibility two is that a valid bitstream is not detected after the zero run within the number of samples specified by the Unknown Timeout Control Register. When this happens the decoding continues, but the input is no longer manually replaced by zeros; i.e. decoding never stops but some initial PCM data is lost. Exit possibility three is that a valid bitstream is detected after the zero run. When this happens the Source Program Status Register is changed from PCM to Unknown and the system re-enters autodetection, upon which it is expected that the bitstream will be redetected and bitstream decoding will begin; i.e. the initial frame or frames of the bitstream are lost.

Both the Zero Run Trigger and Zero Run Restart Control Registers are 32-bit unsigned integers in units of stereo samples with maximum values of 0x7FFFFFFF.

Alpha code for use with the Zero Run Trigger Control Register is given in Table 3-10 (Alpha Code for Audio Input Zero Run Trigger).

Alpha code for use with the Zero Run Restart Control Register is given in Table 3-11 (Alpha Code for Audio Input Zero Run Restart).

TABLE 3-10 **Alpha Code for Audio Input Zero Run Trigger**

Register	Alpha Code	Description
ZeroRun	readIBZeroRun	Return Zero Run Register value. A return of wroteIBZeroRunYes means that this feature is enabled, while wroteIBZeroRunNo means it is disabled.
Zero Run Trigger	readIBZeroRunTrigger	Return Zero Run Trigger Control Register value as described in Section 3.1.10 (Audio Input Zero Run Control).
	writeIBZeroRunTriggerN(NN)	Set the Zero Run Trigger Control Register.
	writeIBZeroRunTriggerN(100)	Select default Zero Run Trigger.

TABLE 3-11 **Alpha Code for Audio Input Zero Run Restart**

Register	Alpha Code	Description
Zero Run Restart	readIBZeroRunRestart	Return Zero Run Restart Control Register value as described in Section 3.1.10 (Audio Input Zero Run Control).
	writeIBZeroRunRestartN(NN)	Set the Zero Run Restart Control Register.
	writeIBZeroRunRestartN(100)	Select default Zero Run Restart.

3.1.11 Audio Input Mode Control

Some applications require special handling of zero runs. The mode control register is used to select amongst a few pre-defined handling configurations.

TABLE 3-12 **Alpha Code for Audio Input Mode Register**

Register	Alpha Code	Description
Mode	readIBMode	Returns the contents of the 8-bit mode control register.
	writeIBModeN(NN)	Set the Mode register.
	writeIBModeN(0)	Select the default Mode register value.

The IB Mode Control Register setting of 0 indicates default operation.

The IB Mode Control Register setting of 1 indicates that the Zero Run Restart value is ignored but the Zero Run Trigger value is used for entering the zero run state. This may be useful for the Z-topology where the secondary input needs to be decoded even if the primary input is in a Zero Run state (but only if the Continuous Decoder Mode is not used, since it requires IB Mode to be 0). However, the input still transitions to the zero run state when the Zero Run Trigger threshold has been met.

When the IB Mode Control Register is 2, the system never enters the zero run state regardless of the Zero Run Restart and Zero Run Trigger values. Therefore, as long as the stream is locked, the PCM decoder will run and will not lose any samples. However the readIBZeroRun status register remains valid. Note that in this IB mode, if stream is changed to bitstream (non-PCM) then it may generate a brief noise if the bitstream doesn't start from the beginning of the frame as then the first broken frame would be played as PCM due to lack of the missing sync headers. If the bitstream starts from the boundary of the frame without missing the sync headers then there won't be any noise. Also if the bitstream is DTSCD then it will always generate a brief noise while switching.

3.1.12 Audio Input Sample Rate Control

Alpha code for use with the Sample Rate Control Register is given in Table 3-13 (Alpha Code for Audio Input Sample Rate).

TABLE 3-13 **Alpha Code for Audio Input Sample Rate**

Register	Alpha Code	Description
Sample Rate Data	readIBSampleRateData	Returns the contents of the 8-bit SampleRateData status register which indicates the sample rate information present in the input stream; if available.
	wroteIBSampleRateDataUnknown	Unknown SampleRateData.
	wroteIBSampleRateDataNone	No SampleRateData.
	wroteIBSampleRateDataNNNNNHz	NNNNN Hz SampleRateData, where NNNNN is one of: 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 64000, 88200, 96000, 128000, 176400, 192000.
Sample Rate Measured	readIBSampleRateMeasured	Returns the contents of the 8-bit SampleRateData-Measured status register which indicates the physical sample rate of the input stream.
	wroteIBSampleRateMeasuredUnknown	Unknown SampleRateMeasured.
	wroteIBSampleRateMeasuredNone	No SampleRateMeasured.
	wroteIBSampleRateMeasuredNNNNNHz	NNNNN Hz SampleRateMeasured, where NNNNN is one of: 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 64000, 88200, 96000, 128000, 176400, 192000.
Sample Rate Status	readIBSampleRateStatus	Returns the contents of the 8-bit SampleRateDataStatus status register which indicates the sample rate associated with the input stream.
	wroteIBSampleRateStatusUnknown	Unknown SampleRateStatus.
	wroteIBSampleRateStatusNone	No SampleRateStatus.
	wroteIBSampleRateStatusNNNNNHz	NNNNN Hz SampleRateStatus, where NNNNN is one of: 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 64000, 88200, 96000, 128000, 176400, 192000.

Register	Alpha Code	Description
Sample Rate Override	readIBSampleRateOverride	Returns the contents of the 8-bit SampleRateOverride control register. This register controls the source of the SampleRateStatus register contents. If this register is 0x80 or 0x82 then the SampleRateStatus register contains the SampleRateMeasured value. If this register is 0x81 then the SampleRateStatus register contains the SampleRateData value. Otherwise the SampleRateStatus register contents contain the SampleRateOverride value.
	writeIBSampleRateOverrideUnknown	Writes to the 8-bit SampleRateOverride register and forces the SampleRateStatus value to UNKNOWN.
	writeIBSampleRateOverrideNone	Writes to the 8-bit SampleRateOverride register and forces the SampleRateStatus value to NONE.
	writeIBSampleRateOverrideNNNNNHz	Writes to the 8-bit SampleRateOverride register and forces the SampleRateStatus value to NNNNNHz, where NNNNN is one of: 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 64000, 88200, 96000, 128000, 176400, 192000.
	writeIBSampleRateOverrideStandard	Writes to the 8-bit SampleRateOverride register with the value 0x80. Forces the SampleRateStatus register to contain the SampleRateMeasured value.
	writeIBSampleRateOverrideData	Writes to the 8-bit SampleRateOverride register with the value 0x81. Forces the SampleRateStatus register to contain the SampleRateData value.
	writeIBSampleRateOverrideMeasured	Writes to the 8-bit SampleRateOverride register with the value 0x82. Forces the SampleRateStatus register to contain the SampleRateMeasured value.
	writeIBSampleRateOverrideAuto	Writes to the 8-bit SampleRateOverride register with the value 0x83. This functionality is not currently supported and shouldn't be used.
	writeIBSampleRateOverrideOther	Writes to the 8-bit SampleRateOverride register with the value 0x84. This functionality is not currently supported and shouldn't be used.

Register	Alpha Code	Description
Scan at High Sample Rate	readIBScanAtHighSampleRateMode	Returns the contents of the 8-bit ScanAtHighSampleRateMode control register. If autodetection is applied the input stream, e.g. via writeDECSourceSelectAuto, then this register controls the application of autodetection when the SampleRateStatus register value is greater than 48000Hz. If the ScanAtHighSampleRateMode is ScanAtHighSampleRateModeDisable (default) then autodetection will not be applied. If the ScanAtHighSampleRateMode is ScanAtHighSampleRateModeEnable then autodetection will be applied.
	writeIBScanAtHighSampleRateModeDisable	Writes to the 8-bit ScanAtHighSampleRateMode control register and disables the application of autodetection, when applicable, to input streams whose SampleRateStatus is greater than 48000Hz.
	writeIBScanAtHighSampleRateModeEnable	Writes to the 8-bit ScanAtHighSampleRateMode control register and enables the application of autodetection, when applicable, to input streams whose SampleRateStatus is greater than 48000Hz.

3.1.12.1 More on ‘Supported’ and ‘Recognized’ Sample Rates

Sample rates identified in Table 3-13 on page 47 as potential values for the ‘NNNNN’ in wroteIBSampleRateDataNNNNNHz (i.e., 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 64000, 88200, 96000, 128000, 176400, 192000) are ‘recognized’. Here, ‘recognized’ means that PA is capable of attaching an identification of the sample rate to the audio data, so that the potential exists for the data to be processed as appropriate for that sample rate.

Not all ‘recognized’ sample rates are actually ‘supported’ by all algorithms currently, for a number of reasons, including:

- support for additional sample rates may require allocation of additional memory for filter coefficients
- some algorithms licensed from third-parties are not licensed for some sample rates
- some algorithms incur excessive processing loads at higher sample rates

The Appendix for each algorithm identifies any sample rate limitations associated with that algorithm.

3.1.13 DTS16 as DTS for Large Sample Rate

The DTS16 as DTS for Large Sample Rate Register controls the reporting of the DTS 16-bit input format (`wroteDECSourceProgramDTS16`) vs. DTS for DVD (`wroteSourceProgramDTS`). The default (`writeIBreportDTS16AsDTSForLargeSampleRateDisable`) reports the source program as detected by inspecting the input data bitstream without regards to the sample rate. When enabled (`writeIBreportDTS16AsDTSForLargeSampleRateEnable`), if the measured sample rate of the input bitstream is greater than 44.1 kHz, then DTS16 is reported as DTS DVD.

TABLE 3-14 **Alpha Code for DTS16 as DTS for Large Sample Rate Register**

Register	Alpha Code	Description
DTS16 as DTS for Large Sample Rate	readIBreportDTS16AsDTSForLargeSampleRate	Returns the contents of the 8-bit mode control register.
	writeIBreportDTS16AsDTSForLargeSampleRateDisable	Disable the DTS16 as DTS for Large Sample Rate register.
	writeIBreportDTS16AsDTSForLargeSampleRateEnable	Enable the DTS16 as DTS for Large Sample Rate register.

see Information Manual, Issue 5.

3.2 Audio Decode

Material regarding the AAC, AC3 and DTS decoders in this section is not applicable to Feature Setsince not include any bitstream decodersAudio decode describes the processing realized by the Decode Algorithm itself. Audio decode is located in the Performance Audio Stream immediately following audio input and is shown shaded in the following figures for each topology.

PAI: Figure 3-6 (Location of the Audio Decode Component for PAI)

PAY: Figure 3-7 (Location of the Audio Decode Component for PAY)

PAD: Figure 3-8 (Location of the Audio Decode Component for PAD)

PAH: Figure 3-9 (Location of the Audio Decode Component for PAH)

PAZ: Figure 3-10 (Location of the Audio Decode Component for PAZ)

FIGURE 3-6 Location of the Audio Decode Component for PAI

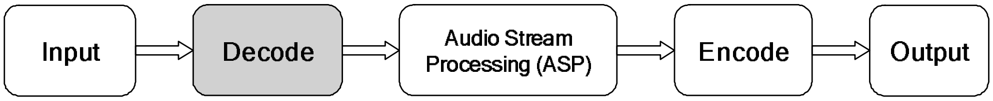


FIGURE 3-7 Location of the Audio Decode Component for PAY

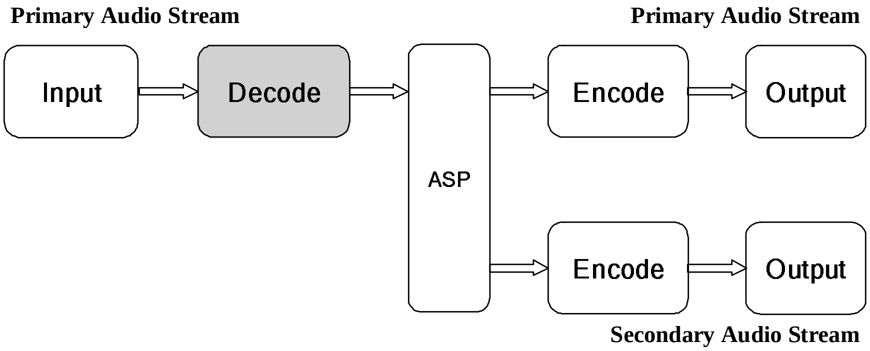


FIGURE 3-8 Location of the Audio Decode Component for PAD

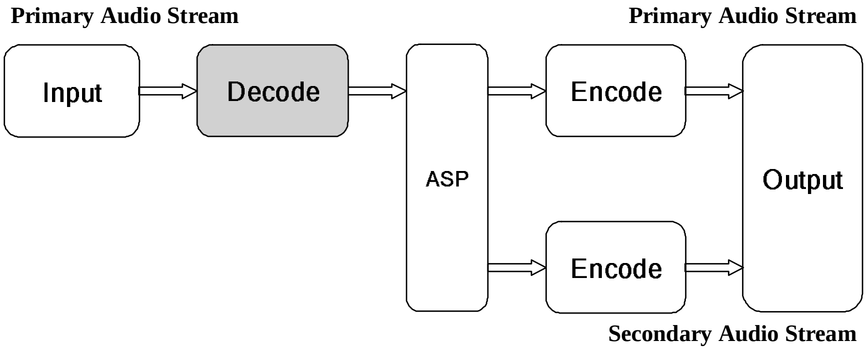


FIGURE 3-9 **Location of the Audio Decode Component for PAH**

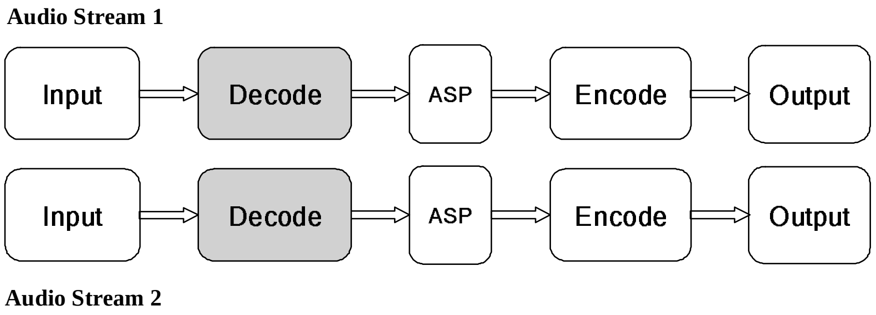
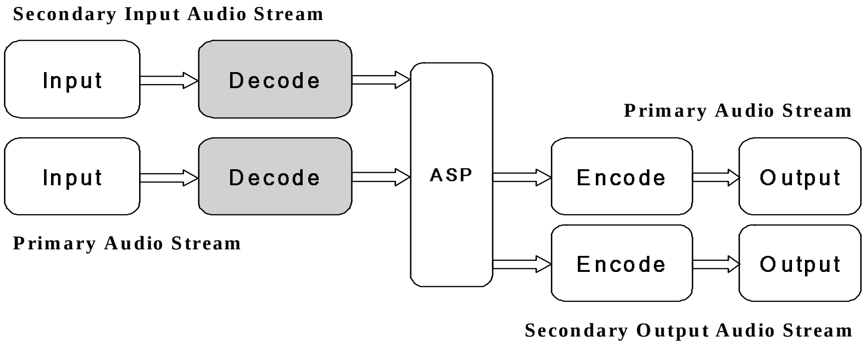


FIGURE 3-10 **Location of the Audio Decode Component for PAZ**



Alpha code for use with the control and status registers for audio decode is given in Table 3-15 (Alpha Code for Audio Decode). Default settings are shown with a shaded background.

TABLE 3-15 **Alpha Code for Audio Decode**

Register	Alpha Code[a]	Description
Sample Rate	readDECSampleRate	Indicate input sample rate as described in TBD.
	wroteDECSampleRate*X*	The numerical values returned are PA/F-standard sample rate values as given in TBD.

Register	Alpha Code[a]	Description
Source Select[b]	readDECSourceSelect	Return current setting for the Decoder Selection mechanism.
	writeDECSourceSelectUnknown	
	writeDECSourceSelectNone	
	writeDECSourceSelectPass	
	writeDECSourceSelectSing	Select Signal/Noise Generator algorithm and ignore input bitstream
	writeDECSourceSelectAuto	System detects input source type and assigns appropriate decoder automatically.
	writeDECSourceSelectBitStream	Select automatic source determination for non-PCM input.
	writeDECSourceSelectDTSAll	System detects input source type and assigns DTS decoder for all DTS source. It includes DTS DVD and DTS CD source including DTSHD streams.
	writeDECSourceSelectPCMAuto	System detects input source type and plays only PCM source. This source select mode will not play noise for Non-PCM input. For Non-PCM input, silence will be generated.
	writeDECSourceSelectPCM	Force PCM decoder to be used with all inputs. This source select mode will play Non-PCM input also as PCM and hence may cause noise to be played if the input bitstream is not PCM.
	writeDECSourceSelectPC8	
	writeDECSourceSelectAC3	Force AC3 decoder to be used with all inputs.
	writeDECSourceSelectDTS	Force DTS DVD decoder to be used with all inputs. This source select mode will not play DTS CD
	writeDECSourceSelectAAC	Select AAC source.
	writeDECSourceSelectMPEG	Select MPEG source.
	writeDECSourceSelectDTS12	
	writeDECSourceSelectDTS13	
	writeDECSourceSelectDTS14	Force DTS CD decoder to be used with all inputs. This source select mode will not play DTS DVD
	writeDECSourceSelectDTS16	
	writeDECSourceSelectWMP	Select WMA Pro source
	writeDECSourceSelectMP3	Select MPEG1 Layer 3 source
	writeDECSourceSelectDSD1	
	writeDECSourceSelectDSD2	
	writeDECSourceSelectDSD3	
	writeDECSourceSelectDDP	Force DDP decoder to be used with all inputs.
	writeDECSourceSelectDTSHD	Force DTSHD decoder to be used with all inputs.
	writeDECSourceSelectTHD	Force TrueHD decoder to be used with all inputs.
	writeDECSourceSelectDXP	Force DTS Express decoder to be used with all inputs.
	writeDECSourceSelectWMA	Select WMA source.

Register	Alpha Code[a]	Description
Source Program	readDECSourceProgram	Indicate source program type as described in Section 5.1.6 (Audio Input Source Modes). Indicate the type of input source that is currently detected.
	wroteDECSourceProgram*X*	The symbolic values returned are PA/F-standard source values as given in Table 3-17 (Alpha Code for Source Registers).
Source Decode	readDECSourceDecode	Indicate which decoder is active.
Channel Configuration Program	readDECChannelConfiguration-Program	Indicate input channel configuration as described in Section 3.2.3 (Audio Decode Channel Configuration).
	wroteDECChannelConfiguration-Program*X*	The symbolic values returned are PA/F-standard channel configuration values as given in Table 3-20 (Alpha Code for Audio Decode Channel Configuration Registers).
Program Format	readDECProgramFormat	Indicate input program format as described in Section 3.2.4 (Audio Decode Program Format).
	wroteDECProgramFormat	The 32-bit numerical value returned is a bit-mapped program format as described in Table 3-22 (Decode Program Format Status Register Bit Map).

a. Default settings are shown with a shaded background.

b. Despite its name, this is a control register and not a select register.

3.2.1 Audio Decode Sample Rate

*Material regarding the AAC, AC3 and DTS decoders in this section applicable only to Feature Set 11 and 13. It is not applicable to Feature Set 14 as it does not include these decoders.*This section provides basic information about the sample rate at the output of the decoder.

PA provides information about the sample rate of the decoded audio data and about the sample rate at which all subsequent processing and output likely takes place within the system.

Decode Sample Rate Status Register

The Decode Sample Rate Status Register indicates the sample rate of the decoded audio data, as determined by the Decode Algorithm. Different Decode Algorithms may determine this sample rate in different ways:

- The PCM Algorithm determines the audio decode sample rate, described in Section 3.1.5 (Audio Input Sample Rate), as read from the input bit stream for digital input. For analog input, the PCM Algorithm sets the audio decode sample rate as per the user's request.
- The AAC Algorithm determines the audio decode sample rate as read from the input bit stream.
- The AC3 Algorithm determines the audio decode sample rate as read from the input bit stream.
- The DTS Algorithm determines the audio decode sample rate as read from the input bit stream under control of decode options which control the production of 88.2 kHz or 96 kHz output from a 44.1 kHz or 48 kHz bit stream.
- The SNG Algorithm uses 48kHz as the audio decode sample rate.

As given in Table 3-15 (Alpha Code for Audio Decode), send alpha code `readDECSampleRate` to determine the sample rate determined by the Decode Algorithm. The response will be one of the values indicated in Table 3-16 (Alpha Code for Decode Sample Rate Register).

TABLE 3-16 **Alpha Code for Decode Sample Rate Register**

Symbol	Description
wroteDECSampleRateUnknown	Sample rate is not known but is treated as 48 kHz.
wroteDECSampleRateNone	Sample rate is not known and blocks processing.
wroteDECSampleRate*F*Hz	Sample rate is *F* Hz. *F* is one of the values given in the second column of TBD
wroteDECSampleRate	Sample rate is something else.

For more information on this status register and other control and status registers related to the sample rate of audio input, see Section 3.1.5 (Audio Input Sample Rate) and Section 3.2.1 (Audio Decode Sample Rate).

3.2.2 Audio Decode Source Modes

*Material regarding the AAC, AC3, DTS and MPEG decoders in this section is not applicable to Feature Set 14 as it does not include any of these decoders.*The source material presented at the input for decoding and processing is known as the *program*. Information about the source program is provided in this section, which describes use of the following registers shown in Table 3-15 (Alpha Code for Audio Decode):

- Decode Source Select Control Register
- Decode Source Program Status Register
- Decode Source Decode Status Register

The common format of the alpha code for these Source Registers is provided in Table 3-17 (Alpha Code for Source Registers). From these, the specific alpha codes for the registers mentioned above are obtained as follows:

- replace *XXX* by *DEC, and*
- replace *Register* by *Select*, *Program*, or *Decode*.

For example, replacing *XXX* by *DEC*, and *Register* by *SourceSelect*,

- alpha code `writeDECSourceSelectAuto` indicates that the type of input source is automatically determined.

As another example, replacing *XXX* by *DEC*, and *Register* by *SourceProgram*,

- alpha code `wroteDECSourceProgramAC3` indicates that the source is Dolby Digital.

TABLE 3-17 **Alpha Code for Source Registers**

Symbol	Description
write*XXX*Source*Register*Unknown	The source is unknown.
write*XXX*Source*Register*None	There is no source.
write*XXX*Source*Register*Pass	The source is connected directly to the sink.
write*XXX*Source*Register*Sing	The source is a generator.
write*XXX*Source*Register*Auto	The source is automatically determined as some specific type given below.
write*XXX*Source*Register*BitStream	The source is automatically determined as some specific type given below other than PCM.
write*XXX*Source*Register*PCM	The source is PCM.
write*XXX*Source*Register*AC3	The source is Dolby Digital.
write*XXX*Source*Register*DTS	The source is DTS DVD IEC Type 11.
write*XXX*Source*Register*AAC	The source is AAC.
write*XXX*Source*Register*MPEG	The source is MPEG.
write*XXX*Source*Register*DTS12	The source is DTS DVD IEC Type 12.
write*XXX*Source*Register*DTS13	The source is DTS DVD IEC Type 13.
write*XXX*Source*Register*DTS14	The source is DTS CD in a 14-bit encoding.
write*XXX*Source*Register*DTS16	The source is DTS CD in a 16-bit encoding.
write*XXX*Source*Register*DDP	The source is Dolby Digital Plus.
write*XXX*Source*Register*THD	The source is Dolby TrueHD
write*XXX*Source*Register*DTSHD	The source is either DTS Master Audio or High Resolution Audio
write*XXX*Source*Register*DXP	The source is DTS Express
write*XXX*Source*Register*DSD1	The source is DSD1.
write*XXX*Source*Register*DSD2	The source is DSD2.
write*XXX*Source*Register*DSD3	The source is DSD3.
write*XXX*Source*Register*MP3	The source is MP3.
write*XXX*Source*Register*WMP	The source is WMA Pro.

Decode Source Select Control Register

The Decode Source Select Control Register is used to select either automatic detection of input bit-stream type or direct selection of a specific decoding algorithm to use with the input bit stream:

- Alpha code `writeDECSourceSelectAuto` automatically determines the type of input source. Automatic Source Select is designed to be used with a digital source. This mode will automatically detect what type of bit stream is being input and select the appropriate decoding method automatically.

  This mode is recommended over the specific modes so as to avoid user error. This is the default setting.
- Bit Stream Source Select is designed to be used with a digital source. This mode will automatically detect what type of bit stream is being input and select the appropriate decoding method automatically. This automatic detection does not include PCM decoding.

  Bit Stream Source Select is recommended when the source is known to be an encoded bit stream other than PCM.
- Various direct forms of source select can be used with a digital source. It is possible to specify Dolby Digital, DTS, AAC, or PCM decoding directly if the topology in use supports that source type.

  In the case of DTS, one of 5 source select values must be used rather than one as shown in Table 3-18 (DTS Source Select Symbols). It is not possible to select DTS decoding in general without naming the DTS encoding format in specific.

Note: *It should be pointed out that when switching from a Direct form of source select to the Automatic form of source select, the switch is a deferred control operation (deferred control operations are discussed in detail in Section 5.4.1 (Understanding Control Operation Deferral)). This means that the switch back to the Automatic source selection will not occur until either the Decoder state machine is restarted via* `writeDECCommandRestart` *or the audio restarting (see Section 5.4.3 (Overcoming Control Operation Deferral)).*

An example of this behavior can be illustrated when providing a PCM input while in the Direct form of source select (via `writeDECSourceSelectPCM`*) and subsequently switching to the Automatic form of source select (via* `writeDECSourceSelectAuto`*). Unless the Decoder state machine is restarted (via* `writeDECCommandRestart`*), even if the input changes from PCM to a bitstream, the PCM decoder will be utilized. This may result in distorted output.*

- Alpha code `writeDECSourceSelectPCM` tells the system that the source is PCM. PCM Source Select is appropriate for use with an analog source, or with a 2-channel, non-encoded digital source, such as a music CD.
- None Source Select is used for test purposes when no source is desired.
- Pass Source Select is used for test purposes to bypass the Performance Audio Framework and directly connect the SIO Input Device to the SIO Output Device.[14]

[14] This capability is currently not supported.

TABLE 3-18 **DTS Source Select Symbols**

Symbol	Description
DTS	DTS DVD IEC Type 11
DTS12	DTS DVD IEC Type 12
DTS13	DTS DVD IEC Type 13
DTS14	DTS CD 14-bit
DTS16	DTS CD 16-bit
DTSHD	DTS High Resolution and master Audio

Decode Source Program Status Register

- The Decode Source Program Status Register indicates the type of input source that has been detected, or selected. The detection feature is particularly useful when Automatic or Bit Stream Source Select is in use. If PA is set to Automatic or Bit Stream Source Select, the program status changes based upon the input bit stream. This register will reflect the change in input bit stream at the moment that such a determination is made. If PA is set to Automatic or Bit-Stream Source Select, the source program status changes based upon the input bit stream. This register reflects the change in input bit stream type at the moment that such a determination has been definitively made.
- If PA is not set to Automatic or Bit-Stream Source Select, the Decode Source Program Status Register will reflect the program determination that the active Decode Algorithm has made rather than actually determining what the source is.

The Decode Source Program Status Register can take on any of the values listed in Table 3-17 (Alpha Code for Source Registers), with the following caveats:

- Neither Source Program Auto nor Bit Stream is ever indicated.
- Source Program Unknown is indicated if the framework determines that no other classification is suitable. This indication is provided by alpha code `writeDECSourceProgramUnknown`.
- A bit-stream type may be recognized by Automatic Source Select, but a valid decoder may not exist.

  For example, this register may indicate `writeDECSourceProgramMPEG` even though an MPEG decoder does not exist in PA.

  In the case of Source Program DTS, one of 5 source program values is used as shown in Table 3-18. A DTS source program in general is not indicated.
- Source Program PCM may be indicated for both analog and digital sources. Alpha code `writeDECSourceProgramPCM` indicates that the source program is PCM.
- Source Program Sing or None is indicated if selected.

Decode Source Decode Status Register

The Source Decode Status Register indicates which decoder is being used at any given moment, a feature that is particularly useful when Automatic or Bit-Stream Source Select is in use. If PA is set to Automatic or Bit-Stream Source Select, the decoder in use changes based upon the input bit stream. This register will reflect the change in decoders at the moment that the switch occurs.

The Source Decode Status Register can take on any of the values listed in Section 3.1.3 (Audio Input Source Modes) for the Source Program Status Register.

If PA is not set to Automatic or Bit-Stream Source Select, this register will reflect the Decode Algorithm selection that has been made directly.Additional alpha codes are given and described in Table 3-19 (Alpha Code for Audio Decode Source Modes).

TABLE 3-19 **Alpha Code for Audio Decode Source Modes**

Register	Alpha Code	Description
Source Select[a]	readDECSourceSelect	Return decoder type that the Source Select Control Register is set to. [b]
	writeDECSourceSelectAuto	Select automatic source determination.
	writeDECSourceSelect*X*	The symbolic values used are PA/F-standard source values as given in Table 3-17 (Alpha Code for Source Registers).
Source Decode	readDECSourceDecode	Indicate source decode type.
	wroteDECSourceDecode*X*	The symbolic values returned are PA/F-standard source values as given in Table 3-17 (Alpha Code for Source Registers).
Source Dual	readDECSourceDual	Return current setting for dual mono source control.
	writeDECSourceDualStereo	Reproduce dual mono source as stereo, with channel 1 on left and channel 2 on right.
	writeDECSourceDualMono1	Reproduce channel 1 of dual mono source as mono.
	writeDECSourceDualMono2	Reproduce channel 2 of dual mono source as mono.
	writeDECSourceDualMonoMix	Reproduce dual mono source as mono, mixing channels 1 and 2.

Register	Alpha Code	Description
Source Karaoke	readDECSourceKaraoke	Return current operation setting for Karaoke.
	writeDECSourceKaraokeAware	Operate decoder as karaoke aware.
	writeDECSourceKaraokeCapableUser	Operate decoder as karaoke capable with user-defined mixing coefficients.
	writeDECSourceKaraokeCapableNone	Operate decoder as karaoke capable with system-defined mixing coefficients to realize reproduction of no vocals.
	writeDECSourceKaraokeCapableLeft	… vocal channel 1.
	writeDECSourceKaraokeCapableRght	… vocal channel 2.
	writeDECSourceKaraokeCapableBoth	… both vocal channels.
Karaoke Capable User	readDECKaraokeCapableUserQ6	Return Decode Karaoke Capable User Control Register values.
	readDECKaraokeCapableUser*X*Q6	Return Decode Karaoke Capable User Control Register values, based on *X*, where *X* is one of Vocal1Level, Vocal1Pan, Vocal2Level, Vocal2Pan, MelodyLevel, or MelodyPan
	writeDECKaraokeCapableUserQ6N6(*NN0*,*NN1*,*NN2*,*NN3*, *NN4*,*NN5*)	Set all Decode Karaoke Capable User Control Registers simultaneously.
	writeDECKaraokeCapableUserVocal1LevelQ6N(*NN*)	Set karaoke capable user vocal 1 level to NN, where NN is an 8-bit Q6 value[c].
	writeDECKaraokeCapableUserVocal1PanQ6N(*NN*)	Set karaoke capable user vocal 1 pan to NN, where NN is an 8-bit Q6 value.
	writeDECKaraokeCapableUserVocal2LevelQ6N(*NN*)	Set karaoke capable user vocal 2 level to NN, where NN is an 8-bit Q6 value.
	writeDECKaraokeCapableUserVocal2PanQ6N(*NN*)	Set karaoke capable user vocal 2 pan to NN, where NN is an 8-bit Q6 value.
	writeDECKaraokeCapableUserMelodyLevelQ6N(*NN*)	Set karaoke capable user melody level to NN, where NN is an 8-bit Q6 value.
	writeDECKaraokeCapableUserMelodyPanQ6N(*NN*)	Set karaoke capable user melody pan to NN, where NN is an 8-bit Q6 value.

[a] Despite its name, this is a control register and not a select register. Control and select registers are described in Section 7.1.1 (Control and Status Registers).

[b] Despite its name, this is a control register and not a select register.

c. *Q6 format* means there are six bits to the right of the “binary point.” This gives resolution to the nearest 0.015625 units (2^{-6}). Since this is an 8-bit value, there are 2 bits to the left of the binary point. One is the sign bit, which leaves 1 bit of magnitude to the left of the point. This gives a representable range of [-2.0,2.0), which allows the value 1.0 to be represented but not the value 2.0. This terminology is discussed in detail in the Application Report: *The Theory and Practice of Volume*.

Source Dual Control

The Source Dual Control Register selects the manner in which dual mono input is reproduced as described in Table 3-19.

Source Karaoke Control

The Source Karaoke Control Register selects the manner in which karaoke input is reproduced as described in Table 3-19.

Example 3-1 (Source Mode) demonstrates the usage of the registers covered in this section.

EXAMPLE 3-1

Source Mode

*This example is applicable only to Feature Sets 11 and 13.This example is not applicable to Feature Set 14 as it doesnot include either the Dolby Digital or DTS decoders.*This example demonstrates the usage of the Source Select Control Register and the Source Program and Source Decode Status Registers.

Send the appropriate Digital Input IOS alpha shortcut for ex execPAIInDigital. Connect a Dolby Digital source to the digital input of DA830 EVM and start the Dolby Digital bit stream as required by the source device, *e.g.*, using the play button.

View the default value of the Source Select Control Register using alpha code `readDECSourceSelect`. The response to this read command should be alpha code `writeDECSourceSelectAuto`.

View the Source Program and Decode Status Registers using alpha code `readDECSourceProgram, readDECSourceDecode`. The response should be `wroteDECSourceProgramAC3, wroteDECSourceDecodeAC3` indicating that the source program is AC3, and also that the AC3 decoder has been selected. Stop the Dolby Digital bit stream, *e.g.*, using the stop or pause button, and again send the read commands. Now, the response should be alpha code `wroteDECSourceProgramUnknown, wroteDECSourceDecodeNone`.[15]

Force PCM decoding by using alpha code `writeDECSourceSelectPCM`. Play the Dolby Digital bit stream. The output will be noisy, since the Dolby Digital bit stream is being interpreted as PCM data.

Again view the Source Program and Decode Status Registers using alpha code `readDECSourceProgram, readDECSourceDecode`. The response should be `wroteDECSourceProgramAC3, wroteDECSourceDecodePCM`[16] indicating that even though the source program is AC3, the PCM decoder is selected. Stop the Dolby

[15] This behavior is desired but may not yet be realized. The actual response obtained may be `wroteDECSourceProgramUnknown, wroteDECSourceDecodeUnknown`.

Digital bit stream, *e.g.*, using the stop or pause button, and again send the read commands. The response should be alpha code `wroteDECSourceProgramUnknown, wroteDECSourceDecodeNone`.[17]

3.2.3 Audio Decode Channel Configuration

PA provides information regarding the audio channels present in the source material. This section specifically describes the alpha codes used with the Decode Channel Configuration Program Status Register in Table 3-15 (Alpha Code for Audio Decode).

Decode Channel Configuration Program

The Decode Channel Configuration Program Status Register indicates the channel configuration of the program at the input to the Decode Algorithm. Various parameters are extracted from the source bit-stream to determine which channels are present. Many different channel configurations are possible, and are listed in Table 3-20 (Alpha Code for Audio Decode Channel Configuration Registers). The alpha code symbol names for this register follow the channel configuration common format described in Section 4.5.1.2 (Channel Configuration Common Format).

Send alpha code `readDECChannelConfigurationProgram` to determine the channel configuration of the program at the input to the Decode Algorithm. The response will be `wroteDECChannelConfigurationProgram`*`X`*, with *X* replaced by the actual channel configuration as given in Table 3-20.

TABLE 3-20 **Alpha Code for Audio Decode Channel Configuration Registers**

Symbol	Description[a]
readDECChannelConfigurationProgram	Requests the channel configuration of the program at the input to the Decode Algorithm.
wroteDECChannelConfigurationProgramUnknown	The channel configuration is unspecified.
wroteDECChannelConfigurationProgramNone	There is no channel configuration, and audio data is carried on no channels.
wroteDECChannelConfigurationProgramMono	The channel configuration is mono, carried on the Center Channel.
wroteDECChannelConfigurationProgramStereo	The channel configuration is stereo, carried on the Left and Right Channels.
wroteDECChannelConfigurationProgramStereoLtRt	The channel configuration is surround-encoded stereo, carried on the Left and Right Channels.
wroteDECChannelConfigurationProgramStereoMono	The channel configuration is mono, carried on the Left and Right Channels.

[16] This behavior is desired but may not yet be realized. The actual response obtained may be `wroteDECSourceProgramPCM, wroteDECSourceDecodePCM`.

[17] This behavior is desired but may not yet be realized. The actual response obtained may be `wroteDECSourceProgramPCM, wroteDECSourceDecodePCM`.

Symbol	Description[a]
wroteDECChannelConfigurationProgram3Stereo	The channel configuration is surround carried on the Left, Center, Right, and LFE or Subwoofer Channels.
wroteDECChannelConfigurationProgramPhantom	The channel configuration is surround carried on the Left, Right, Surround, and LFE or Subwoofer Channels.
wroteDECChannelConfigurationProgramSurround	The channel configuration is surround carried on the Left, Center, Right, Surround, and LFE or Subwoofer Channels.
wroteDECChannelConfigurationProgramNone_0	No satellite or subwoofer audio data.
wroteDECChannelConfigurationProgramMono_0	Satellite audio data on Center Channel. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramPhantom0_0	Satellite audio data on Left and Right Channels. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramPhantom0Stereo_0	Satellite audio data on Left and Right Channels that is not surround-encoded. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramPhantom0LtRt_0	Satellite audio data on Left and Right Channels that is surround-encoded. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramPhantom0Mono_0	Satellite audio data on Left and Right Channels that is identical. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramPhantomDual0_0	Satellite audio data on Left and Right Channels that is dual mono. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramPhantom1_0	Satellite audio data on Left, Right, and one Surround Channel. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramPhantom2_0	Satellite audio data on Left, Right, and Surround Channels. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramPhantom2Stereo_0	Satellite audio data on Left, Right, and Surround Channels, where the Surround Channels are not surround-encoded. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramPhantom2LtRt_0	Satellite audio data on Left, Right, and Surround Channels, where the Surround Channels are surround-encoded. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramPhantom2Mono_0	Satellite audio data on Left, Right, and Surround Channels, where the Surround Channels are identical. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramPhantom3_0	Satellite audio data on Left, Right, Surround, and one Back Channel. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramPhantom4_0	Satellite audio data on Left, Right, Surround, and Back Channels. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramSurround0_0	Satellite audio data on Left, Center, and Right Channels. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramSurround1_0	Satellite audio data on Left, Center, Right, and one Surround Channel. No LFE or subwoofer audio data.

Symbol	Description[a]
wroteDECChannelConfigurationProgramSurround2_0	Satellite audio data on Left, Center, Right, and Surround Channels. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramSurround2Stereo_0	Satellite audio data on Left, Center, Right, and Surround Channels, where the Surround Channels are not surround-encoded. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramSurround2LtRt_0	Satellite audio data on Left, Center, Right, and Surround Channels, where the Surround Channels are surround-encoded. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramSurround2Mono_0	Satellite audio data on Left, Center, Right, and Surround Channels, where the Surround Channels are identical. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramSurround3_0	Satellite audio data on Left, Center, Right, Surround, and one Back Channel. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgramSurround4_0	Satellite audio data on Left, Center, Right, Surround, and Back Channels. No LFE or subwoofer audio data.
wroteDECChannelConfigurationProgram*Type*_1	Same as above, but with LFE or subwoofer audio data on one channel rather than on no channels.
wroteDECChannelConfigurationProgramHL(HI16,LO16)	Channel configuration given as a 32-bit numerical value.

[a] If shaded, this indicates that this is a *colloquial symbol* that has the same value as, and takes precedence over in most contexts, a later symbol in the standard form. The symbols in the standard form are given for completeness.

Information regarding the channel configurations of the audio data throughout the decoding process is given in Table 3-21 (Alpha Code for Audio Decode Channel Configurations). An explanation of the alpha codes used with the following registers from Table 3-21 is provided in this section, augmented with symbols from Table 4-3 (Alpha Code for Channel Configuration Registers):

- Channel Configuration Request
- Channel Configuration Decode
- Channel Configuration Downmix
- Channel Configuration Override

TABLE 3-21 **Alpha Code for Audio Decode Channel Configurations**

Register	Alpha Code	Description
Channel Configuration Request	readDECChannelConfigurationRequest	Indicate channel configuration (downmix) requested of the Decode Algorithm. Is controlled by Speaker Control if Speaker Control is enabled.
	writeDECChannelConfigurationRequest*X*	Request a channel configuration for the output of the decoder. See Table 4-3 (Alpha Code for Channel Configuration Registers). Can only be used if Speaker Control is disabled. Overridden if Speaker Control is enabled.
Channel Configuration Request Auxiliary	readDECChannelConfigurationRequestAux	Indicate auxiliary channel configuration (downmix) requested of the Decode Algorithm. Is part of DECChannelConfigurationRequestRegister.
	writeDECChannelConfigurationRequestAuxUnknown	Request unknown auxiliary channel configuration for the output of the decoder.
	writeDECChannelConfigurationRequestAuxStereo	Request stereo auxiliary channel configuration for the output of the decoder.
	writeDECChannelConfigurationRequestAuxLtRt	Request LtRt auxiliary channel configuration for the output of the decoder.
	writeDECChannelConfigurationRequestAuxMono	Request Mono auxiliary channel configuration for the output of the decoder.
	writeDECChannelConfigurationRequestAuxDual	Request Dual Mono auxiliary channel configuration for the output of the decoder.
Channel Configuration Decode	readDECChannelConfigurationDecode	Indicate channel configuration resulting from decode internal to the Decode Algorithm.
	wroteDECChannelConfigurationDecode*X*	See Table 4-3 (Alpha Code for Channel Configuration Registers).
Channel Configuration Downmix	readDECChannelConfigurationDownmix	Indicate channel configuration resulting from downmix internal to the Decode Algorithm.
	wroteDECChannelConfigurationDownmix*X*	See Table 4-3 (Alpha Code for Channel Configuration Registers).
Channel Configuration Override	readDECChannelConfigurationOverride	Indicate channel configuration override of the Decode Algorithm. Is not available for configuration if Speaker Control is enabled.
	writeDECChannelConfigurationOverride*X*	See Table 4-3 (Alpha Code for Channel Configuration Registers).

Decode Channel Configuration Request

The Decode Channel Configuration Request Select Register is the primary mechanism by which the desired channel configuration at the conclusion of decode and Audio Stream Processing is requested. Its value always controls the operation of the Decode Algorithm. Under certain cases, its value controls the operation of the Audio Stream Processing Algorithms as well in the default state, but not under all control register settings. See “Decode Channel Configuration Override” on page 67.

This register is a select register. Thus, it cannot be set by the user while Speaker Control is enabled. In this case, it operates as a status register. If Speaker Control is disabled, then the user may set it directly, as it operates as a control register.

Any of the settings shown in Table 4-3 (Alpha Code for Channel Configuration Registers) are applicable, whether the Decode Channel Configuration Request Select Register is set directly by the host or whether it is set indirectly via the user. Most of these settings have straightforward meanings as regards the requested channel configuration. One setting, however, has a special meaning as follows:

- If the Decode Channel Configuration Request Select Register is set to Unknown, this signals the Decode Algorithm that it should generate its output according to the Decode Channel Configuration Program Status Register value.

  In this case, no downmix is performed by either the decode or downmix subroutines of the Decode Algorithm, and a minimal amount of Audio Stream Processing will be applied to this result in the generation of the audio stream output.[18]

Decode Channel Configuration Decode

The Decode Channel Configuration Decode Status Register indicates what channel configuration has resulted from operation of the decode subroutine of the Decode Algorithm in response to the Channel Configuration Request. The audio data described by this channel configuration does not exist in any form that is accessible, but this status is a very important and valuable piece of information that describes one step of the audio decoding process.

Decode Channel Configuration Downmix

The Decode Channel Configuration Downmix Status Register indicates what channel configuration has resulted from operation of the downmix subroutine of the Decode Algorithm in response to the Channel Configuration Request. The audio data described by this channel configuration is passed to the Audio Stream Processing. Thus, this status register describes the channel configuration of the audio data ultimately produced by the Decode Algorithm.

[18] This Audio Stream Processing will not be nil, since algorithms such as Bass Management will continue to be applied, but no de-emphasis, surround processing, *etc*. will be applied since the channel configuration is already that requested. This is the ultimate first, best fit, since the requested and current channel configurations are identical.

Decode Channel Configuration Override

The Decode Channel Configuration Override Select Register is a secondary mechanism by which the desired channel configuration at the conclusion of Audio Stream Processing, but not decode, is achieved. Its value controls the operation of the Audio Stream Processing Algorithms except, as in the default state, when the value is Unknown. In this case, the value of the Decode Channel Configuration Request Select Register is used for this purpose. See "Decode Channel Configuration Request" on page 66.

This register is a select register. Thus, it cannot be set by the user while Speaker Control is enabled. In this case, it operates as a status register. If Speaker Control is disabled, then the user may set it directly, as it operates as a control register.

This register typically operates as a status register apropos the host, controlled by the Audio System. It may be operated as a control register with respect to the host if its use as a control register by the Audio System is disabled.

Any of the settings shown in Table 4-3 (Alpha Code for Channel Configuration Registers) are applicable, whether the Decode Channel Configuration Override Select Register is used as a control register and set directly by the host or whether it is used as a status register and set indirectly via the audio system. Most of these settings have straightforward meanings as regards the requested channel configuration. One setting, however, has a special meaning as follows:

- If the Decode Channel Configuration Override Select Register is set to Unknown, this signals the Audio Stream Processing that it should generate its output according to the Decode Channel Configuration Request Select Register value.

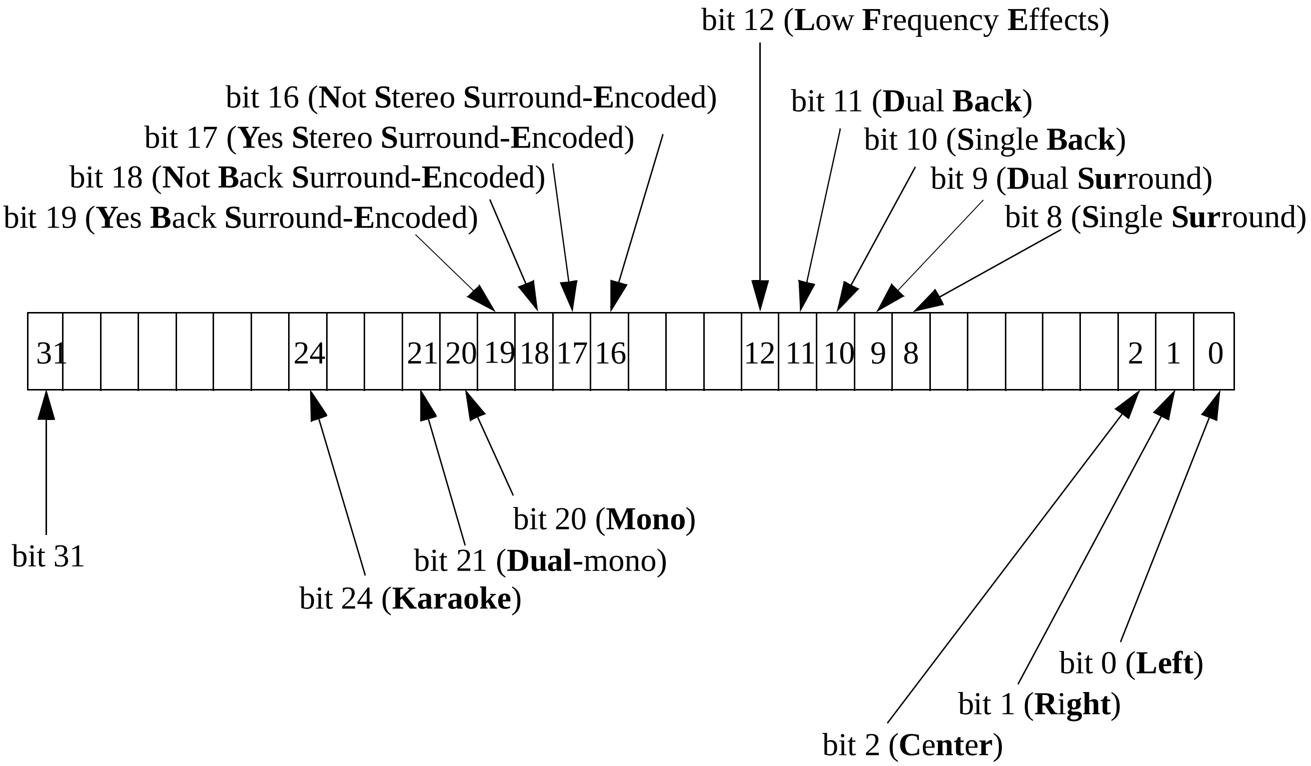
Note that this setting is the default. It results in normal system operation.

3.2.4 Audio Decode Program Format

The Decode Program Format Status Register indicates which channels in the received bit stream contain program information, along with other information such as surround-encoded status for 2-channel inputs.[19] It is a bit-mapped register. Table 3-22 (Decode Program Format Status Register Bit Map) and Figure 3-11 (Decode Program Format Status Register) show how the bits in this register should be interpreted.

[19] The Decode Program Format Status Register is often referred to as simply the Program Format Status Register.

FIGURE 3-11 **Decode Program Format Status Register**



In addition to the basic channel information, the Decode Program Format Status Register contains the following special indicators:

Bits 16 and 17

- A two-channel input bitstream can be marked in one of three ways:
  - Encoded for surround sound matrix processing (Lt/Rt).
  - Not-encoded for surround sound matrix processing (Lo/Ro).
  - No indication as to whether or not it is encoded for surround sound processing (Lu/Ru).

  Two bits are used to indicate one of these three cases. If the Yes Stereo Surround-Encoded Bit is set, the bitstream is encoded for surround sound (Lt/Rt). If the Not Stereo Surround-Encoded Bit is set, the bitstream is not encoded for surround sound (Lo/Ro). If neither bit it set, the bitstream has no indication of whether or not it is encoded for surround sound (Lu/Ru). At no time will both bits be set.

Bits 18 and 19

- A multi-channel input bitstream can be marked in one of three ways:
  - Encoded for back channel processing (Lt/Rt).
  - Not-encoded for back channel processing (Lo/Ro).

- No indication as to whether or not it is encoded for back channel processing (Lu/Ru).

Two bits are used to indicate one of these three cases. If the Yes Back Channel - Encoded Bit is set, the bitstream is encoded for Back Channel processing (Lt/Rt). If the Not Back Channel-Encoded Bit is set, the bitstream is not encoded for Back Channel processing (Lo/Ro). If neither bit it set, the bitstream has no indication of whether or not it is encoded for Back Channel processing (Lu/Ru). At no time will both bits be set.

Bit 20

- The Mono Bit indicates that the program is monophonic, either true mono or dual mono reproduced as mono. The Dual Bit indicates if the program is dual mono. In the case of dual mono input, both the Mono and Dual bits may be set. See Table 3-22.

Bit 24

- The Karaoke Bit indicates that the program is karaoke.

For example, if the input source is a stereo PCM bit stream, this register will show that the main (left and right) channels contain program material and, if the PCM Program Channel Configuration Control Register so specifies, that this program is not stereo surround-encoded. Dolby Digital, AAC, and DTS bit streams can contain program material on as few as one channel or as many as eight.

TABLE 3-22 **Decode Program Format Status Register Bit Map**

Bit	Name	Program	Indication
0	Left	Multi-channel	Left Channel
		Mono	Mono Channel in conjunction with Right
		Dual Mono	Dual Mono Channel 1
1	Rght	Multi-channel	Right Channel
		Mono	Mono Channel in conjunction with Left
		Dual Mono	Dual Mono Channel 2
2	Cntr	Multi-channel	Center Channel
		Mono	Mono Channel
		Dual Mono	Dual Mono Channel Mix
8	SSur		Single Surround Channel
9	DSur		Dual Surround Channels[a]
10	SBak		Single Back Channel
11	DBak		Dual Back Channels[b]
12	LFE		Low Frequency Effects Channel
16	NSSE		Not Stereo Surround-Encoded Indicator
17	YSSE		Yes Stereo Surround-Encoded Indicator
18	NBSE		Not Back Surround-Encoded Indicator
19	YBSE		Yes Back Surround-Encoded Indicator

Bit	Name	Program	Indication
20	Mono		Mono Indicator[c]
21	Dual		Dual-Mono Indicator[d]
24	Karaoke		Karaoke Indicator
other			Reserved[e]

[a] This bit should be used to drive both the LSur and RSur indicators.

[b] This bit should be used to drive both the LBak and RBak indicators.

[c] This bit will be set if the program material is mono, and it may be set if the program material is dual mono. It is set if any of the Left, Rght, or Cntr Bits are used to indicate the reproduction of mono or dual mono rather than multi-channel program material.

[d] This bit will be set if the program material is dual mono. It is reset otherwise.

[e] Reserved bits should indicate zero, but may not in special circumstances.

Strictly speaking, the Decode Program Format Status Register does not simply report the *program format*, but rather the *program reproduction format*. This is because reproduction information for mono and dual mono is included in the indicators. The information for dual mono is included as per section 4.11.8 “Dual Mono” of the Dolby Digital Licensee Information Manual, Issue 5. The information for mono is included to provide equivalent indication for these cases.

Certification Requirement:

> *Dolby requires indicators that provide information as to what audio format is present in an input bit stream. This register can be used to drive these indicators directly. See section 4.11.6 “Program Format Display for Multi-channel Products” of the* Dolby Digital Licensee Information Manual, Issue 5.
>
> *Dolby requires indicators that provide information as to what form of reproduction is used for an input dual-mono bit stream. This register can be used to drive these indicators directly. See section 4.11.8 “Dual Mono” of the* Dolby Digital Licensee Information Manual, Issue 5.[20]

Some of the alpha codes for audio decode are given in Table 3-23 (Alpha Code for Audio Decode) and described below. Additional alpha codes are given and described in the rest of this section.

- The Status Register in Table 3-23 is a *status* register whose contents can be read by a *read* command. Alpha code `readDECStatus` returns the entire DEC status structure.
- The Mode Register in Table 3-23 is a *control* register that can be written to by a *write* command, as well as its contents can be read by a *read* command. This register can be used to enable or disable decode operation.Use of the Command Action Register, Command Result Register, and Command Register from Table 3-23 is covered in Section 10.2 (Audio Stream Processing (Advanced)).
- No other information is provided for the Frame Count Register and the Frame Length Register other than that given in Table 3-23.

[20] This capability is not provided by the Decode Program Format Status Register. Such capability would require a “reproduction format register,” which PA does not provide.

TABLE 3-23 **Alpha Code for Audio Decode**

Register	Alpha Code	Description
Status	readDECStatus	Return entire DEC status structure.
Control	readDECControl	Return entire DEC status structure.
Mode	readDECMode	Return Decode Mode Control Register value.
	writeDECModeDisable	Disable decode operation.
	writeDECModeEnable	Enable decode operation.
	writeDECModeContinuous[a]	Enable continuous decode operation for secondary input.
Command	readDECCommand	Return Decode Command Register value, including both action and result portions.
	writeDECCommandNone	See Section 10.2 (Audio Stream Processing (Advanced)) and Table 7-2 (Alpha Code for Audio Decode Audio Operation Coordination).
	writeDECCommandAbort	
	writeDECCommandRestart	
Frame Count	readDECFrameCount	Indicate the number of input frames processed by the Decode Algorithm in this bit stream.
	writeDECFrameCountHL(HH,LL)	Write the number of input frames to be processed by the Decode Algorithm in this bit stream. Currently not available.
	wroteDECFrameCount	The number of input frames is given as a 32-bit unsigned integer.
Frame Length	readDECFrameLength	Indicate the frame length produced by the Decode Algorithm.
	wroteDECFrameLength	The frame length is given as a 16-bit unsigned integer.
Bypass	readDECBypass	Return Decoder Bypass Control register value. The implementation of the bypass operation is dependent on the decoder algorithm. Currently only PCM decoder supports this feature. Please refer to the PCM Decoder Appendix for details of the supported bypass operation.
	writeDECBypassDisable	Disable bypass operation.
	writeDECBypassEnable	Enable bypass operation. Please note as currently only PCM decoder supports the bypass functionality, it is recommended that the decoder source select be set to PCM when this functionality is enabled. This will ensure that only PCM decoder operates when the bypass operation is enabled.

a. writeDECModeContinuous is the default for non-”None” secondary input IOS Shortcuts (e.g., with Z-topology).

3.2.5 Audio Decode Pre-emphasis

PA provides information about the presence of pre-emphasis in the decoded audio data.

The Decode Emphasis Select Register indicates whether or not the input bit stream had pre-emphasis filtering applied, and therefore whether de-emphasis filtering should be applied to the decoded audio data. Alpha code for use with this select register is given in Table 3-24 (Alpha Code for Audio Decode Pre-emphasis).

TABLE 3-24 **Alpha Code for Audio Decode Pre-emphasis**

Register	Alpha Code	Description
Emphasis	readDECEmphasis	Return Decode Emphasis Select Register value. Whether this value represents the detected pre-emphasis of the input bit stream is dependent on the Input Buffer Emphasis Override Control Register setting.
	wroteDECEmphasisNo	Input bit stream did not have pre-emphasis filtering applied.
	wroteDECEmphasisYes	Input bit stream had pre-emphasis filtering applied. De-emphasis filtering should be applied to the decoded audio data.

The value indicated by this register is dependent on the setting of the Input Buffer Emphasis Override Control Register, as described in Section 3.1.4 (Audio Input Pre-Emphasis).

3.2.6 Audio Decode Channel Map

The Decode Channel Map To Control Register is used to determine the manner in which the output channels of the Decode Algorithm are mapped onto the audio channels presented for Audio Stream Processing. It is used in conjunction with the Decode Channel Map From Control Register described in Section 3.1.8 (Audio Input Channel Map) in the manner discussed in Section 4.3.1 (Channel Maps).

This mapping is very flexible. It allows any of the *N* decode-output channels to be mapped onto any of the *M* audio channels. In addition, any of the *N* decode-output channels can be replaced with a zero-valued audio channel during its mapping to an audio channel, and a zero-valued decode-output channel is available for mapping onto one of the *M* audio channels irrespective of any decode-output channel.

The Decode Channel Map To Control Register consists of 16 8-bit sub-registers, of which only the first 8 are pertinent. By convention, all 16 sub-registers are set simultaneously with a single alpha code as shown in Table 3-25 (Alpha Code for Audio Decode Channel Map), even though only 8 are actually used. Each sub-register can take on one of two types of values as shown in Table 3-26 (Decode Channel Map To Values), either a control number represented by negative integers or a channel number indicated by non-negative integers. Data is provided for the channel corresponding to the number of that sub-register at the decode input according to the value in that sub-register.

TABLE 3-25 **Alpha Code for Audio Decode Channel Map**

Register	Alpha Code	Description
Channel Map To	readDECChannelMapTo	Return Decode Channel Map To Control Register value.
	writeDEChannelMapTo16(-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3)[a]	Select PA/F-standard decode channel map.
	writeDECChannelMapTo16(…)	Select alternate decode channel map. This macro takes 16 arguments.
	wroteDECChannelMapTo,...	Used to indicate the Decode Channel Map To Control Register value.

[a] The default value is as shown for the primary audio stream but otherwise for the secondary audio stream.

Examples best serve to illustrate the operation of the Decode Channel Map To Control Registers:

- Consider the default, PA/F-standard decode channel map from Table 3-25 (Alpha Code for Audio Decode Channel Map), reproduced here:

  writeDECChannelMapTo16(0,1,8,9,2,12,10,11,3,4,5,6,7,13,14,15)

  This causes the decode-output channels generated by the Decode Algorithm to be mapped onto the audio stream in the PA/F-standard order as shown in Table 3-27 (Multi-Channel Decode/Encode Channel Map).
- Non-standard use of the decode channel map to is possible but has significant consequences:
  1. Standard Audio Stream Processing Algorithms rely upon this PA/F-standard channel ordering and, thus, cannot be used.
  2. A corresponding encode channel map is required. See Section 4.1.5 (Audio Encode Channel Map).

  Non-standard usage of the decode channel map to will be quite rare.
- In all cases, only the values of the first N_d Decode Channel Map From Control Registers are pertinent, for N_d defined as in Section 3.1.8 (Audio Input Channel Map). This means, for example, that for the input of PA, the last 8 arguments shown below are arbitrary:

  writeDECChannelMapTo16(...,3,4,5,6,7,13,14,15)

  By convention, these values are used in all cases, but such usage is by convention only and is not required.

For more information on decode channel maps, see Section 3.1.8 (Audio Input Channel Map).

TABLE 3-26 **Decode Channel Map To Values**

Value[a]	Description
< -1	No audio data is provided for this channel. Here "< -1" means any number less than -1; e.g., -2 or -3.
-1	Zero-valued audio data is provided for this channel.
PAF_LEFT	Audio data is presented from this channel to the Left Channel if this channel is present in the decode channel mask.
PAF_RGHT	Audio data is presented from this channel to the Right Channel if this channel is present in the decode channel mask.
…	…
PAF_SUBW	Audio data is presented from this channel to the Subwoofer Channel if this channel is present in the decode channel mask.

[a] The symbolic values are as given in `paftyp_a.h`.

TABLE 3-27 **Multi-Channel Decode/Encode Channel Map**

Logical Audio Channel	Physical Audio Channel[a]
Left	PAF_LEFT
Rght	PAF_RGHT
LSur	PAF_LSUR
RSur	PAF_RSUR
Cntr	PAF_CNTR
Subw	PAF_SUBW
LBak	PAF_LBAK
RBak	PAF_RBAK
RWide	PAF_RWID
LWide	PAF_LWID
LHeight	PAF_LHED
RHeight	PAF_RHED

[a] The symbolic values are as given in `paftyp_a.h`.

3.2.7 Audio Decode Buffer Ratio

PA allows control of the percentage of the audio frame buffer utilized by the Decode Algorithm. Alpha code for controlling the buffer ratio is given in Table 3-28 (Alpha Code for Audio Decode Buffer Ratio).

TABLE 3-28 **Alpha Code for Audio Decode Buffer Ratio**

Register	Alpha Code	Description
Buffer Ratio[a]	readDECBufferRatio	Indicate the percentage of the possible audio data buffer frame length produced by the Decode Algorithm.
	writeDECBufferRatio1	Produce frames at 100% of the possible frame length. This setting should be used with writeSRCRateRequestFull.
	writeDECBufferRatio2	Produce frames at 50% of the possible frame length. This setting should be used with writeSRCRateRequestDouble.
	writeDECBufferRatio4	Produce frames at 25% of the possible frame length. This setting should be used with writeSRCRateRequestQuadruple.
	writeDECBufferRatioE	Produce frames at twice the possible frame length. This setting is not available, since by default, 100% of the audio frame buffer is used.
	writeDECBufferRatioC	Produce frames at quadruple the possible frame length. This setting is not available, since by default, 100% of the audio frame buffer is used.

[a] This register is utilized when using the Synchronous Rate Conversion Algorithm as described in Appendix S (Synchronous Rate Conversion Number 1).

By default, 100% of the audio frame buffer is so utilized. This allows output to be produced at maximal efficiency in terms of execution resources (MIPS). It also restricts, however, use of portions of the audio frame buffer for effects processing such as sample rate conversion of audio data to a sample rate for output that is higher than the sample rate of input.

Use of 50% or 25% of the audio frame buffer by the Decode Algorithm is appropriate when up-sampling is to be performed in a ratio of 1:2 or 1:4, respectively.

Note for PAZ: writeDECBufferRatio2/writeDECBufferRatio4 should be sent to the primary as well as the stream joining it to ensure correct operation. The join will not be successful unless the number of audio samples available at ASJ is the same for all the streams to be joined.

PAY: For PAY, Example 5-6 (Primary Audio Stream Synchronous Rate Conversion, Upsampling) demonstrates use of 50% of the audio frame buffer when upsampling in the Primary audio stream.

CHAPTER 4 Application Interface - Audio Encode/ Output

This chapter, along with the previous two chapters, and the next one, introduces the PA Application Programming Interface (API). The API topics covered here primarily deal with the *encode* and *output* components. Basic and afvanced information on the Encode and Output components is provided in this chapter.

4.1 Audio Encode

Audio encode describes the processing realized by the Encode Algorithm. Audio encode is located in the Performance Audio Stream immediately preceding audio output and is shown shaded in the following figures for each topology.:

Strictly speaking, *audio encode* describes the processing realized by the Encode Algorithm, exclusive of that related to volume, and *audio output* describes the processing required to utilize a bit stream produced by the Encode Algorithm for output.

PAI : Figure (Location of the Audio Encode Component for PAI)

PAY: Figure 4-1 (Location of the Audio Encode Components for PAY)

PAD: Figure 4-2 (Location of the Audio Encode Component for PAD)

PAH: Figure 4-3 (Location of the Audio Encode Component for PAH)

PAZ: Figure 4-4 (Location of the Audio Encode Component for PAZ)

Location of the Audio Encode Component for PAI

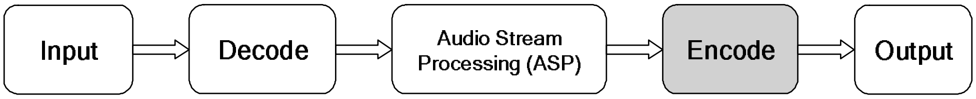


FIGURE 4-1 **Location of the Audio Encode Components for PAY**

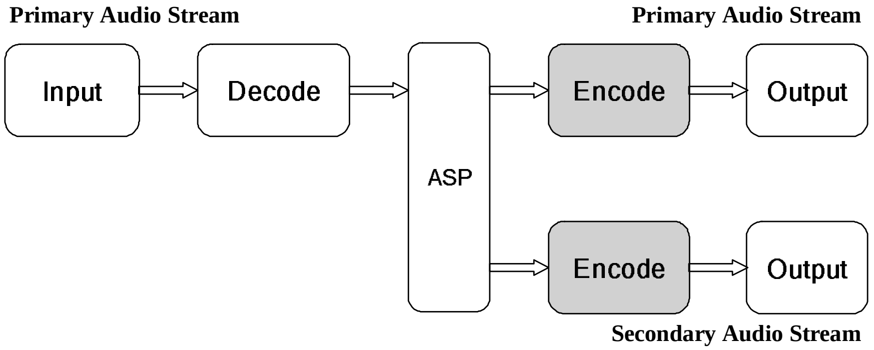


FIGURE 4-2 **Location of the Audio Encode Component for PAD**

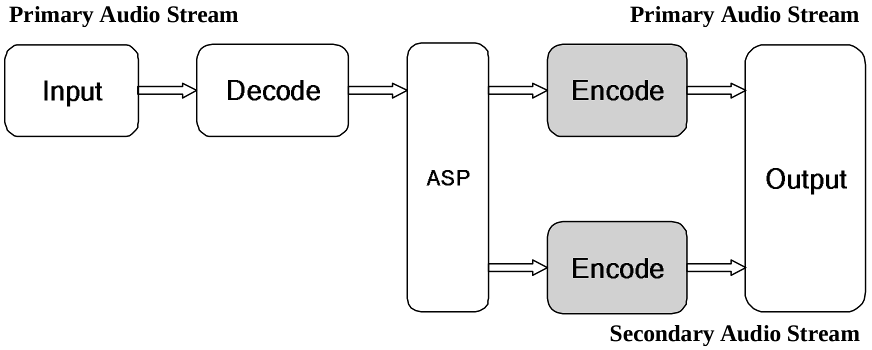


FIGURE 4-3 **Location of the Audio Encode Component for PAH**

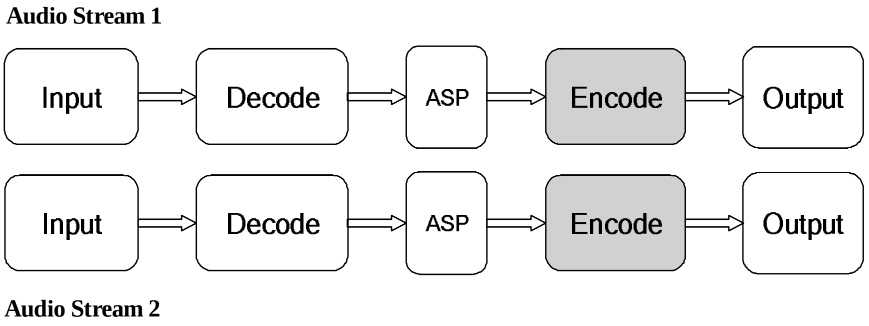
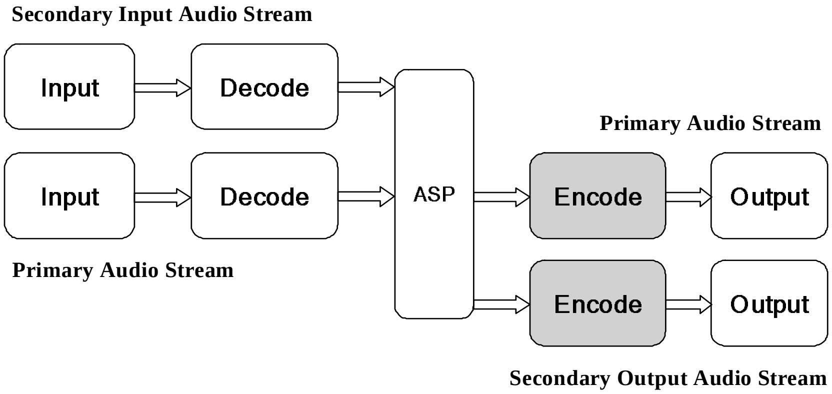


FIGURE 4-4 **Location of the Audio Encode Component for PAZ**



Alpha code for use with the control and status registers for audio encode is given in Table 4-1 (Alpha Code for Audio Encode).

TABLE 4-1 **Alpha Code for Audio Encode**

Register	Alpha Code	Description
Sample Rate	readENCSampleRate	Indicate encode sample rate. The encode sample rate is also the sample rate at the output.
	wroteENCSampleRate*X*	The numerical values returned are PA/F-standard sample rate values as given in Table 4-2 (Alpha Code for Encode Sample Rate Registers).
Listening Format	readENCListeningFormat	Indicate which channels potentially have audio data at the encoder. These same channels also have audio data at the output.
	wroteENCListeningFormat	The 32-bit numerical value returned is a listening format as described in Section 4.1.2 (Audio Encode Listening Format).
Volume Control	Various	See Section 5.4 (Volume Control).

4.1.1 Audio Encode Sample Rate

This section provides basic information about the sample rate at the encoder.

PA provides information about the sample rate of the encoded audio data. The sample rate information provided follows the format as described in Section 2.2.1.1 (Sample Rate Common Format).

As given in Table 4-1 (Alpha Code for Audio Encode), send alpha code `readENCSampleRate` to determine the sample rate at the encoder. The response will be one of the values indicated in Table 4-2 (Alpha Code for Encode Sample Rate Registers).

TABLE 4-2 **Alpha Code for Encode Sample Rate Registers**

Symbol	Description
wroteENCSampleRateUnknown	Sample rate is not known but is treated as 48 kHz.
wroteENCSampleRateNone	Sample rate is not known and blocks processing.
wroteENCSampleRate*F*Hz	Sample rate is *F* Hz. *F* is one of the values given in the second column of Table 2-2 (Sample Rate Symbol Values)
wroteENCSampleRate	Sample rate is something else.

4.1.2 Audio Encode Listening Format

The Encode Listening Format Status Register indicates which channels in the transmitted audio stream contain program information[21], along with other information such as whether or not the channel material is surround-encoded. It is a bit-mapped register. Figure 4-5 (Encode Listening Format Status Register) and Table 4-3 (Encode Listening Format Status Register -- Bitmap) show how the bits in this register should be interpreted.

[21] While these channels do contain program material in the form of audio data, it may be zero-valued audio data.

FIGURE 4-5 **Encode Listening Format Status Register**

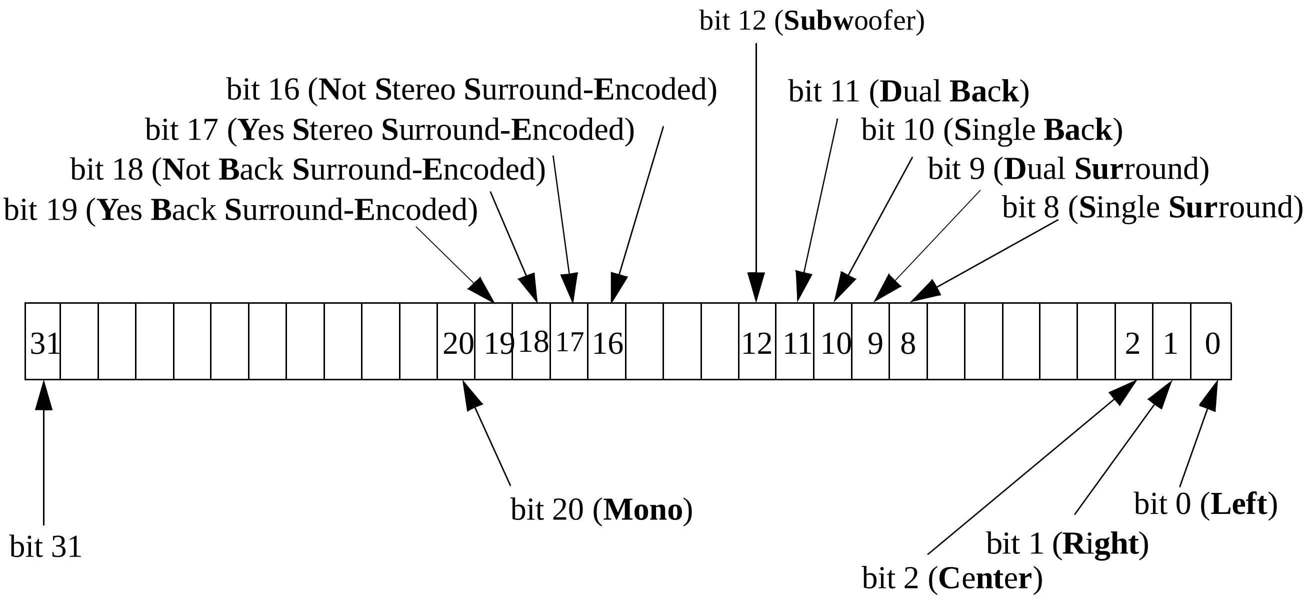

bit 12 (**Sub**w**oofer)
bit 16 (**N**ot **S**tereo **S**urround-**E**ncoded)
bit 11 (**D**ual **Back**)
bit 17 (**Y**es **S**tereo **S**urround-**E**ncoded)
bit 10 (**S**ingle **Back**)
bit 18 (**N**ot **B**ack **S**urround-**E**ncoded)
bit 9 (**D**ual **Sur**round)
bit 19 (**Y**es **B**ack **S**urround-**E**ncoded)
bit 8 (**S**ingle **Sur**round)
31
20
19
18
17
16
12
11
10
9
8
2
1
0
bit 20 (**Mono**)
bit 0 (**Left**)
bit 31
bit 1 (**Right**)
bit 2 (**C**e**nt**e**r**)

TABLE 4-3 **Encode Listening Format Status Register -- Bitmap**

Bit	Name	Program	Indication
0	Left	Multi-channel	Left Channel
1	Rght	Multi-channel	Right Channel
2	Cntr	Multi-channel	Center Channel
		Mono	Mono Channel
8	SSur		Single Surround Channel
9	DSur		Dual Surround Channels[a]
10	SBak		Single Back Channel
11	DBak		Dual Back Channels[b]
12	Subw		Subwoofer Channel
16	NSSE		Not Stereo Surround-Encoded Indicator
17	YSSE		Yes Stereo Surround-Encoded Indicator
18	NBSE		Not Back Surround-Encoded Indicator
19	YBSE		Yes Back Surround-Encoded Indicator
20	Mono		Mono Indicator[c]

[a] This bit should be used to drive both the LSur and RSur indicators.

b. This bit should be used to drive both the LBak and RBak indicators.

c. This bit will be set if the program material is mono, that is, if the Cntr Bit is used to indicate the reproduction of monophonic rather than multi-channel program material.

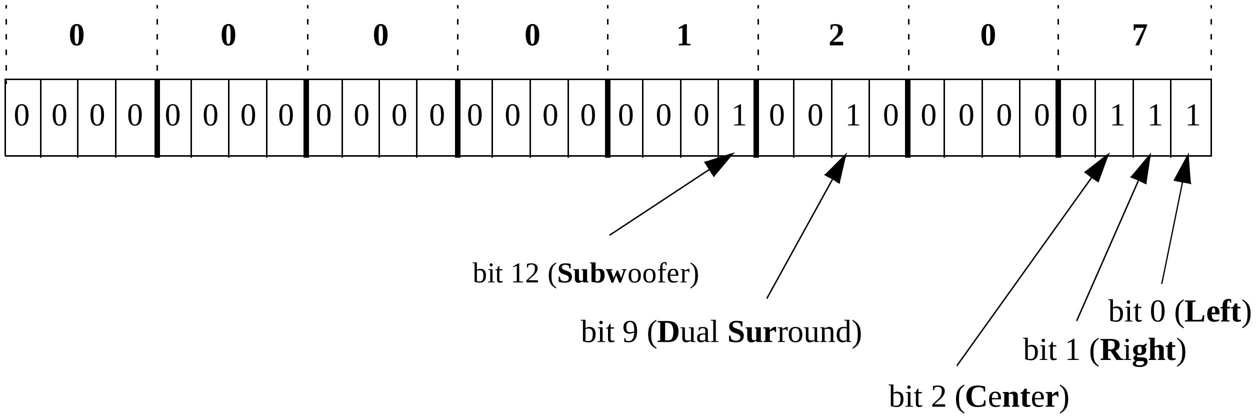
For example, if the output material is a single, stereo PCM bit stream, this register value will be 0x00010003 to show (1) that the main, that is, left and right, output channels contain program material, and (2) that this stereo material is not surround-encoded, that is, that this stereo material is not matrix-encoded as with Dolby Pro Logic IIx, DTS Neo:6 2-Channel Matrix, *etc*.

As a second example, as shown in Figure 4-6 (Encode Listening Format Status Register Example), if the output material is 3/2.1 this register value will be 0x00001207 to show (1) that the front, surround, and subwoofer output channels contain program material, and (2) that the surround material is not back channel-encoded, that is, that this surround material is not matrix-encoded as with Dolby Digital EX, DTS ES Matrix, *etc*.

PCM, Dolby Digital, AAC, and DTS audio streams can produce program material on as few as one channel or as many as eight in the audio stream in PA. This channel count is not indicated via any status register.

FIGURE 4-6 **Encode Listening Format Status Register Example**

Example: 0x00001207 indicates that the front, surround, and subwoofer output channels contain program material.



Certification Requirement:

Dolby requires indicators that provide information as to what output channels an audio signal may be present on. The Encode Listening Format Status Register indicates which channels can be expected to produce an output signal based on the program material in the outgoing bit stream—in other words, the output format selected by the user. This register can be used to drive these indicators. The indicators can thus provide information as to *what consumers can expect regarding the spatial qualities of the output signal.*

Bits 0–15 of the Encode Listening Format Status Register can be used for indicators as to which channels have program material. Bits 16–31 can be used for indicators as to the nature of the program material on these channels. For example, whether or not matrix-encoding of the front or surround channels is indicated in these latter bits of this register.[22] *Note that these values do not necessarily indicate that Dolby Pro Logic IIx is used for the encoding.*

For more information on requirements, see section 4.11.6 "Program Format Display" of the Dolby Digital Licensee Information Manual, Issue 5.

Though the encoder is also responsible for aiding in volume control, this section covers the functions of the encoder exclusive of that related to volume. The topic of Audio Volume is covered separately in Section 5.7 (Volume Control).

Some of the alpha codes for audio encode are given in Table 4-4 (Alpha Code for Audio Encode) and described below. Additional alpha codes are given and described in the rest of this section.

- The Status Register in Table 4-4 is a *status* register whose contents can be read by a *read* command. Alpha code `readENCStatus` returns the entire ENC status structure.
- The Mode Register in Table 4-4 is a *control* register that can be written to by a *write* command, as well as its contents can be read by a *read* command. This register can be used to enable or disable encode operation.
- No other information is provided for the Frame Length Register other than that given in Table 4-4.

TABLE 4-4 **Alpha Code for Audio Encode**

Register	Alpha Code	Description
Status	readENCStatus	Return entire ENC status structure.
Mode	readENCMode	Return Encode Mode Control Register value.
	writeENCModeDisable	Disable encode operation.
	writeENCModeEnable	Enable encode operation.
Frame Length	readENCFrameLength	Indicate the frame length produced by the Encode Algorithm.
	wroteENCFrameLength	The frame length is given as a 16-bit unsigned integer.

[22] Yes X-Surround Encoded bit is set to indicate that the material on channels X is surround-encoded, and No X-Surround Encoded bit is set to indicate that the material on channels X is not surround-encoded. If neither bit is set, this indicates that either such an indication is not known or is not appropriate in the complete context of the program material.

Register	Alpha Code	Description
Bypass	readENCBypass	Return Encoder Bypass Control register value. The implementation of the bypass operation is dependent on the encoder algorithm. Please refer to the PCM Encoder Number 1 and PCM Encoder Number 2 Appendices for details of the supported bypass operation.
	writeENCBypassDisable	Disable bypass operation.
	writeENCBypassEnable	Enable bypass operation.
Command	readENCCommand	Return the ENC Command register.
	writeENCCommandN(N)	Write direct values into the ENC Command register.
	writeENCCommandNone	Write “None” to ENC Command register.
	writeENCCommandMute	Write “Mute” to ENC Command register.
	writeENCCommandUnmute	Write “Unmute” to ENC Command register.

4.1.3 Audio Encode Processes and Formats

PA provides information regarding the format of the material designated for output. See Section 4.1.2 (Audio Encode Listening Format) for details about the Encode Listening Format Status Register. Additional information about the following registers is provided in this section:

- Encode Program Format
- Encode Sample Process

Alpha code for use with these status registers is given in Table 4-5 (Alpha Code for Audio Encode Processes and Formats).

TABLE 4-5 **Alpha Code for Audio Encode Processes and Formats**

Register	Alpha Code	Description
Program Format	readENCProgramFormat	Indicate encode program format.
	wroteENCProgramFormat	The 32-bit numerical value returned is a program format as described in Section 4.1.2 (Audio Encode Listening Format).
Sample Process	readENCSampleProcess	Indicate sample processing, all bytes.
	readENCSampleProcess*N*	Indicate sample processing, byte *N* only.

Encode Program Format

The Encode Program Format Status Register indicates which channels in the transmitted *bit stream* contains program information resulting from this audio stream, along with other information to be indicated.[23] It is a bit-mapped register. Table 4-3 (Encode Listening Format Status Register -- Bitmap) shows how the bits in this register should be interpreted.

By default, the output material is 3/4.1, and this register shows (1) that the front, rear, and subwoofer output channels contain program material, and (2) that this material is not surround-encoded. As an alternate example, if the output material is a stereo PCM bit stream, this register will show (1) that both, that is, left and right, output channels contain program material, and (2) that this material is not surround-encoded.

The Encode Program Format Status Register reports the program format of the encoding in a strict sense. Unlike the Decode Program Format Status Register, it does not include *reproduction* information, only *transmission* information. The Encode Listening Format Status Register, however, does include reproduction information as described below.

PA output always contain program material on eight channels. This channel count is indicated in the Encode Channel Count Status Register.

PAY/PAZ : For PAY/PAZ, the output contains program material on eight primary channels and two secondary channels. This channel count is indicated in the Encode Channel Count Status Register for the Primary and Secondary Audio Stream, respectively.

Encode Sample Process

The Encode Sample Process Status Register passes along *sample process information* generated by ASP Algorithms. It is a bit-mapped register. Table 4-6 (Encode Sample Process Status Register Bit Map) shows how the bits in this register should be interpreted. In interpreting these bits, if set the description applies, and if reset the description does not apply.

TABLE 4-6 **Encode Sample Process Status Register Bit Map**

Byte(s)	Bit(s)	Name	Description
0	0	PL	Dolby Pro Logic is active (applies only to Dolby Pro Logic IIx operating on stereo input).
	1	NEO	NEO is active.
	2	ASA	ASA is active.
	3	DEX	Dolby Surround EX is active (applies only to DEX mode of Dolby Pro Logic IIx, not 6 Channel Music, 7 Channel Movie, or 7 Channel Music modes).
	4	SURRBASS	If set, bass from surround channels is not routed to the subwoofer.
	5	BACKBASS	If set, bass from back channels is not routed to the subwoofer.
	6	MONOBACK	If set, the back channels are mono (correlated even if they are split and routed to two channels).
	7	SURRPROC	If set, an algorithm producing surround channels is active.

[23] While these channels do contain program material in the form of audio data, it may be zero-valued audio data.

Byte(s)	Bit(s)	Name	Description
1	0	BACKPROC	If set, an algorithm producing back channels is active.
	1	PL2X	Dolby Pro Logic IIx is active (applies to all uses of Dolby Pro Logic IIx).
	2	RVB	Room Simulator, Number 1 is active.
	3	MTX	Matrix is active.
	4	GEQ	Graphic Equalizer is active.
	5	SRC	Synchronous Sample Rate conversion is active.
	6	DMX	Downmix is active.
	7	DEM	Deemphasis is active.
2	0–7		Reserved for OEM applications.
3	0–7		Reserved for custom applications.

4.1.4 Audio Encode Channel Configurations

PA allows control over how the audio channels present in the material designated for output are encoded and provides status regarding this encoding. Complete information regarding the channel configurations of the audio data throughout the encoding process is provided in this section.

Alpha code for use with the control and status registers described in this section is given in Table 4-7 (Alpha Code for Audio Encode Channel Configurations) augmented with symbols from Table 2-4 (Alpha Code for Channel Configuration Registers) and Table 2-5 (Alpha Code for Extended Channel Configuration).

TABLE 4-7 **Alpha Code for Audio Encode Channel Configurations**

Register	Alpha Code	Description
Channel Configuration Request	readENCChannelConfigurationRequest	Return Encode Channel Configuration Request Control Register value.
	writeENCChannelConfigurationRequestSurround4_1[a]	Select 8-channel PCM encoding.
	writeENCChannelConfigurationRequest*X*	The numerical values used or returned are PA/F-standard channel configuration values as given in Table 2-4 (Alpha Code for Channel Configuration Registers).
Channel Configuration Stream	readENCChannelConfigurationStream	Indicate channel configuration resulting from audio stream processing.
	wroteENCChannelConfigurationStream*X*	The numerical values returned are PA/F-standard channel configuration values as given in Table 2-4 (Alpha Code for Channel Configuration Registers).

Register	Alpha Code	Description
Channel Configuration Encode	readENCChannelConfigurationEncode	Indicate channel configuration resulting from encoding.
	wroteENCChannelConfigurationEncode*X*	The numerical values returned are PA/F-standard channel configuration values as given in Table 2-4 (Alpha Code for Channel Configuration Registers).

[a] The default is 8-channel PCM encoding for the primary audio stream but 2-channel PCM encoding for the secondary audio stream.

Encode Channel Configuration Request

The Encode Channel Configuration Request Control Register sets the form of the output and controls the number of output channels. The default value shown in Table 4-7 (Alpha Code for Audio Encode Channel Configurations) is appropriate for use with to generate 8-channel PCM output.

This register is used in conjunction with the Encode Channel Map From and To Control Registers. All together, they control exactly how the audio channels are mapped onto the transmit pins as described in Section 4.1.5 (Audio Encode Channel Map) and Section 4.2.8 (Audio Output Channel Map).

Encode Channel Configuration Stream

The Encode Channel Configuration Stream Status Register indicates the form of the output at the conclusion of audio stream processing immediately preceding encoding. That is, it indicates what channel configuration has resulted from operation the Decode and ASP Algorithms in response to various control register settings.

Encode Channel Configuration Encode

The Encode Channel Configuration Encode Status Register indicates the actual form of the output that is generated in response to the request given in the Encode Channel Configuration Request Control Register.[24]

4.1.5 Audio Encode Channel Map

The Encode Channel Map From Control Register is used to determine the manner in which the audio channels which result from Audio Stream Processing are mapped onto the input channels of the Encode Algorithm. It is used in conjunction with the Encode Channel Map To Control Register as described in Section 4.2.8 (Audio Output Channel Map).

This mapping is very flexible. It allows any of the *M* audio channels to be mapped onto any of the *N* channels to be encoded. In addition, any of the *M* audio channels can be replaced with a zero-valued audio channel when mapping from the audio channels irrespective of the encoded channels, and a zero-valued encoded channel is available for mapping to any of the *M* encoded channels irrespective of any audio channel.

[24] Perhaps this value should be modified according to the settings of other registers, like the Encode Channel Map To Control Register, but it is not.

The Encode Channel Map From Control Register consists of 16 8-bit sub-registers, of which only the first 8 are pertinent. By convention, all 16 sub-registers are set simultaneously with a single alpha code as shown in Table 4-8 (Alpha Code for Audio Encode Channel Map), even though only 8 are actually used. Each sub-register can take on one of two types of values as shown in Table 4-9 (Encode Channel Map From Values), either a control number represented by negative integers or a channel number indicated by non-negative integers. Data is provided for the channel corresponding to the number of that sub-register at the input of the Encode Algorithm according to the value in that sub-register.

TABLE 4-8 **Alpha Code for Audio Encode Channel Map**

Register	Alpha Code	Description
Channel Count	readENCChannelCount	Indicate the number of channels in the channel configuration resulting from encoding.
	wroteENCChannelCountN(8)[a]	Eight channels.
Channel Map From	readENCChannelMapFrom	Return Encode Channel Map From Control Register value.
	writeENCChannelMapFrom16(0,1,8,9,2,12,10,11,3,4,5,6,7,13,14,15)[b]	Select PA/F-standard encode channel map.
	writeENCChannelMapFrom16(…)	Select alternate encode channel map. This macro takes 16 arguments.
	wroteENCChannelMapFrom,...	Used to indicate the Encode Channel Map From Control Register value.

a. The default value is 8 for the primary audio stream but 2 for the secondary audio stream.

b. The default value is as shown for the primary audio stream but otherwise for the secondary audio stream.

Examples best serve to illustrate the operation of the Encode Channel Map From Control Register:

- Consider the default, PA/F-standard encode channel map from Table 4-8 (for the Primary Audio Stream in case of PAY), reproduced here:

  writeENCChannelMapFrom16(0,1,8,9,2,12,10,11,3,4,5,6,7,13,14,15)

  This causes the encode-input channels used by the Encode Algorithm to be mapped from the audio stream in the PA/F-standard order as shown in Table 3-34 (Multi-Channel Decode/Encode Channel Map).

- Consider the alternate encode channel map shown here:

  writeENCChannelMapFrom16(0,1,0,1,0,1,0,1,3,4,5,6,7,13,14,15)

  This causes the Left and Right Channels in the audio stream to be duplicated four times across the encode-input channels used by the Encode Algorithm. This creates an “Eight-Channel Stereo PCM Encoding.”

- Consider the alternate encode channel map shown here:

  writeENCChannelMapFrom16(0,0,0,0,0,0,0,0,3,4,5,6,7,13,14,15)

  This causes the Left Channel in the audio stream to be duplicated eight times across the encode-input channels used by the Encode Algorithm. This creates an “Eight-Channel Mono PCM Encoding.”
- In all cases, only the values of the first N_e Encode Channel Map From Control Registers are pertinent, for N_e defined as in Section 4.2.8 (Audio Output Channel Map). This means, for example, that for the output of PA, the last 8 arguments shown below are arbitrary:

  writeENCChannelMapFrom16(...,3,4,5,6,7,13,14,15)

  By convention, these values are used in all cases, but such usage is by convention only and is not required.

It is tempting to use the Encode Channel Map From Control Register to achieve a simple reordering of the encoded channels for output via device pins. Such usage is discouraged, however. Use of the Encode Channel Map To Control Registers is recommended for such purposes as described in Section 4.2.8.

For more information on encode channel maps, see Section 4.2.8 and Section 4.3.1 (Channel Maps).

TABLE 4-9 **Encode Channel Map From Values**

Value[a]	Description
-2	No audio data is provided for this channel.
-1	Zero-valued audio data is provided for this channel.
PAF_LEFT	Audio data is provided for this channel from the Left Channel if this channel is present in the encode channel mask.
PAF_RGHT	Audio data is provided for this channel from the Right Channel 1 if this channel is present in the encode channel mask.
…	…
PAF_SUBW	Audio data is provided for this channel from the Subwoofer Channel if this channel is present in the encode channel mask.

[a] The symbolic values are as given in `paftyp_a.h`.

4.2 Audio Output

Audio output is located at the end of the Performance Audio Stream and is shown shaded in the following figures for each topology

PAI: Figure 4-7 (Location of the Audio Output Component for PAI)

PAY: Figure 4-8 (Location of the Audio Output Components for PAY)

PAD: Figure 4-8 (Location of the Audio Output Components for PAY)

PAH: Figure 4-10 (Location of the Audio Output Components for PAH)

PAZ:

Basic concepts regarding audio output is presented in this section. For more information, see Section 8.5 (Audio Output (Advanced)).

FIGURE 4-7 Location of the Audio Output Component for PAI

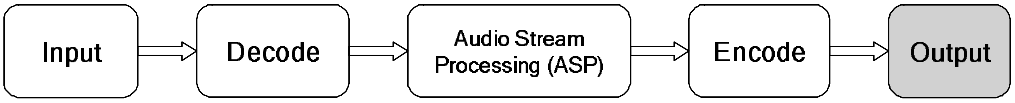


FIGURE 4-8 Location of the Audio Output Components for PAY

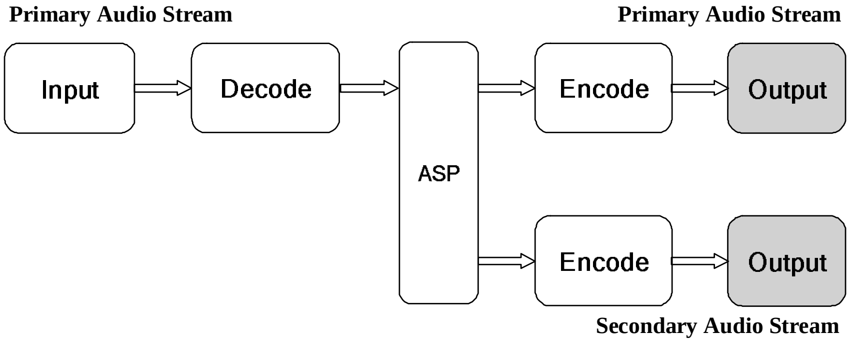


FIGURE 4-9 Location of the Audio Output Components for PAD

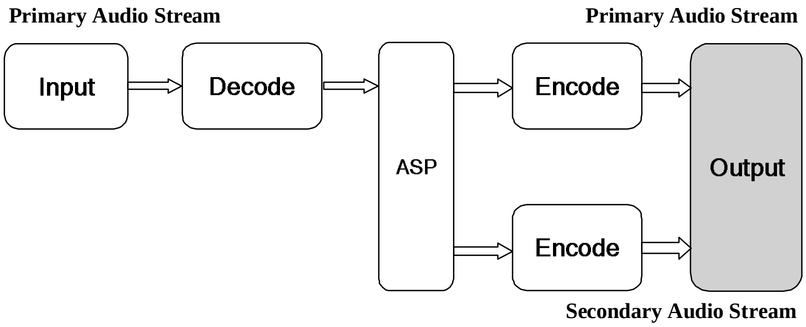


FIGURE 4-10

Location of the Audio Output Components for PAH

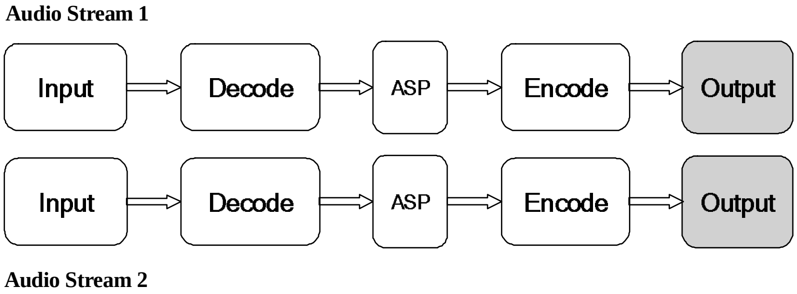
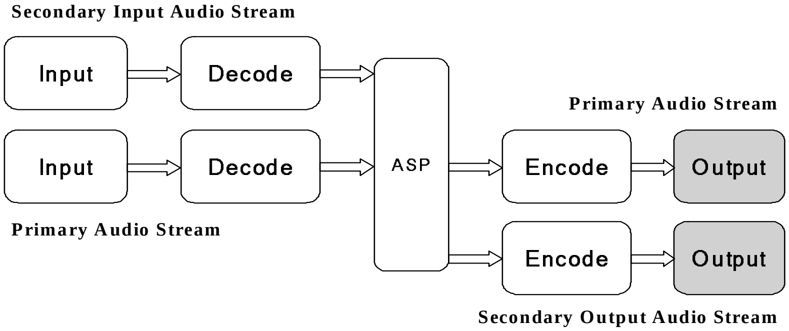


FIGURE 4-11

Location of the Audio Output Components for PAZ



PAY/PAZ: PAY and PAZ provide two audio outputs, the primary output from the primary audio stream and the secondary output from the secondary audio stream.[25] Primarily, the material presented in this section is applicable to both audio streams in an equivalent manner. Where appropriate, a distinction is made below between the operation of the two audio streams. **PAD:** PAD provides a single audio output that includes both the primary audio stream and secondary audio stream. Since a single output is used for both streams, they have the limitation of using the same output sample rate. Additionally, the number of channels are limited in the primary stream to 6 (instead of 8 with PAY).

[25] The primary output is generated by Audio Stream 1, and the secondary output is generated by Audio Stream 2.

4.2.1 Audio Output Sample Rate

The Encode Sample Rate Status Register indicates the sample rate of the audio output. See Section 4.1.1 (Audio Encode Sample Rate) for information about the Encode Sample Rate Status Register.

4.2.2 Audio Output Default

By default no output is produced when hardware is powered on. Specific output IOS alpha codes need to be sent to get desired output.

4.2.3 Audio Output IOS

PA-specific PA Messages allow output to be muted or routed to other connectors . These alpha code symbols cause the invocation of alpha code shortcuts to realize other output modes. The type of output supported on the DA830 hardware are given in Table 4-10 (Audio Output Shortcuts).

TABLE 4-10 **Audio Output Shortcuts**

Topology	Alpha Code	Description
PAI	execPAIOutNone	No output is heard
	execPAIOutAnalog	Output heard after D/A conversion
	execPAIOutDigital	Digital output
PAY	execPAYOutPrimaryNone	No output is heard
	execPAYOutPrimaryAnalog	Output heard after D/A conversion
	execPAYOutDigital	Digital output
	execPAYOutSecondaryNone	No output is heard
	execPAYOutSecondaryDigital	Digital Output

These IOS Modes are realized using a sequence of alpha code commands that result in the following basic control register settings:

- Output Buffer SIO Select Command Register: This register is set as appropriate.

Other control register settings are engaged, but these are beyond the scope of this chapter. The following information regarding the alpha code symbols shown in is pertinent:

- If a requested IOS Mode is available, the identical execution command is returned.
- If a requested IOS Mode is not available, the request is processed as if it was a request for no output, that is, “None,” and the error value (see below) is returned. This condition can result if, for example, a resource conflict exists (*e.g.*, two, simultaneous uses of a single physical device are requested) or if the hardware cannot support such a required output signal route (*e.g.*, due to).

  The error value is `execDerivedNameInOutError`. It is used to return error status. It should not be used as an execution command.

- If a serious error occurs during processing of the shortcut, the response `0xdead` will be returned. A time-out on the communications link indicates a fatal error within the system.

The following note provides specific information about IOS Hardware Support for PAY and PAZ.

Note: IOS Hardware Support for PAY/PAZ

Other IOS Modes may be supported, but these are not listed here because of the difficulty in documenting this moving target. To determine if a particular IOS Mode is available, try it. If the error return value does not or does result, this indicates that the desired IOS Mode is or is not available, respectively.

4.2.3.1 Muting the System

The `writeENCCommandMute`/`writeENCCommandUnmute` commands () should be used to mute and unmute the system, respectively. If there is no input available, the system will execute the mute/unmute such that once input is played it is either muted or unmuted according to the command sent.

4.2.4 Audio Output Devices

PA provide two kinds of audio output devices that can be accessed as one of “device sets” from PA as follows:[26]

- Digital-Analog Conversion devices.
- Digital Transmitter group, which are accessed alone.

4.2.5 Audio Output Listening Format

The Encode Listening Format Status Register indicates which channels in the transmitted audio stream contain program information, along with other information to be indicated. See Section 4.1.2 (Audio Encode Listening Format) for information about the Encode Listening Format Status Register.

4.2.6 Audio Output Volume

Volume Control is a vital part of the control of audio output. This topic is discussed separately in Section 5.7 (Volume Control).

Output Buffer SIO Select Command Register

[26] Definitions for these devices are provided in the Glossary.

The Output Buffer SIO Select Command Register indicates the audio output device that is currently *selected* for use, or *pending* for use. Alpha code for use with this register is given in and Table 4-11.

TABLE 4-11 **Alpha Code for Audio Output Buffer - II**

Register	Alpha Code	Description
Output Buffer SIO Select[a]	readOBSioSelect	Indicate the output device set selection for this audio stream as described in Section 4.2.4 (Audio Output Devices). For PAY, this is the audio stream at which the alpha code is being directed.
	writeOBSioSelectN(NN)	Select current output device set as from .

[a] Despite its name, this is a command register and not a select register.

Send alpha code `readOBSIOSelect` to determine the output device. The response, as given by the alpha codes in , indicates the device selected. Only the alpha code response corresponding to the device *selected* is given in .

From , it can be seen that the response to `readOBSIOSelect` is of the form `wroteOBSIOSelect`*`Device`* indicating the *Device* selected

The alpha code response corresponding to the device *pending* is similar to that in , except that the word *Select* in the alpha code symbol name is replaced by the word *Command*.

Section 4.2.4 (Audio Output Devices) describes in more detail the various audio input devices that are available with PA.

In Table 4-12 (Alpha Code for Audio Output Buffer - General) the general OB registers are described.

TABLE 4-12 **Alpha Code for Audio Output Buffer - General**

Register	Alpha Code	Description
Status	readOBStatus	Return entire OB status structure.
Control	readOBControl	Returns the control registers of the OB status structure
Mode	readOBMode	Returns the contents of the 8-bit mode control register.
Output Buffer Clock Select	readOBClock	Returns the contents of the 8-bit clock control register.
	writeOBClockInternal	Writes to the 8-bit clock control register with the value 0x00. Indicates that the bit and frame clocks for the output are provided internally.
	writeOBClockExternal	Writes to the 8 bit clock control register with the value 0x01. Indicates that the bit and frame clocks for the output are provided externally and need not be generated by the DSP.

Register	Alpha Code	Description
Output Flush	readOBFlush	Returns the contents of the 8-bit flush control register.
	writeOBFlushDisable	Writes to the 8 bit clock flush register with the value 0x00. At the end of decode processing, e.g. when the input stops, the remaining output data will be truncated.
	writeOBFlushEnable	Writes to the 8 bit clock flush register with the value 0x01. At the end of decode processing, e.g. when the input stops, the remaining output data will be output.

4.2.7 Audio Output Clocks

The contents of this section are applicable only to I topology.

In the typical output modes of PA for digital and analog audio output, the bit and frame clocks used to drive the McASP transmit peripheral are generated internal to that peripheral as derivatives of the master clock of the audio input. This clock generation is quite sophisticated in that it automatically generates transmit clocks appropriate for both standard, single-rate output as well as DTS 96/24, double-rate output. Such *internal transmit clock generation* is preferred for PA:

- *PA Output Digital* uses bit and frame clocks generated internal to the DSP to drive the McASP transmit peripheral.

These typical output modes are designed for use in real systems.

4.2.8 Audio Output Channel Map

The Encode Channel Map To Control Register is used to determine the manner in which the output channels of the Encode Algorithm are mapped onto the channels of the SIO Output Device. It is used in conjunction with the Encode Channel Map From Control Register described in Section 4.1.5 (Audio Encode Channel Map) in the manner discussed in Section 4.3.1 (Channel Maps).

This mapping is flexible. For each of the SIO channels, either no encode-output channel or any particular encode-output channel can be mapped.

The Encode Channel Map To Control Register consists of 16 8-bit sub-registers, of which only the first 8 are pertinent. While up to 16 sub-registers can be manipulated either singly or in a group, in practice N_e sub-registers are set simultaneously with a single alpha code as shown in Table 4-13 (Alpha Code for Audio Output Channel Map To), where N_e is the number of sub-registers to be used by the Encode Algorithm: $N_e = 2$ for stereo PCM output, $N_e = 8$ for multi-channel PCM output, *etc.* Each sub-register can take on one of two types of values. A negative value signals that no encode-output channel is to be mapped onto this SIO channel by the Encode Algorithm.[27] A non-negative value signals that this encode-output channel is to be mapped onto this SIO channel for by the Encode Algorithm. This construction allows multiple instantiations of the Encode Algorithm to feed one SIO Output Device, allowing multiple output zones via a single SIO Output Device using non-overlapping channel sets.

[27] In this case, the corresponding Encode Channel Map From Control Register value is ignored.

TABLE 4-13 **Alpha Code for Audio Output Channel Map To**

Register	Alpha Code	Description
Channel Map To	readENCChannelMapTo	Return Encode Channel Map To Control Register value.
	writeENCChannelMapTo16(0,4,1,5,2,6,3,7,8,12,9,13,10,14, 11,15)[a]	Select default output channel map for 8-channel output.
	writeENCChannelMapTo2(...)	Select standard output channel map for *N* channels, *N*=2, 8, or 16. These macros take *N* arguments each.
	writeENCChannelMapTo8(…)	
	writeENCChannelMapTo16(…)	
	wroteENCChannelMapTo,...	Used to indicate the Encode Channel Map To Control Register value.

[a] The default value is as shown for the primary audio stream but otherwise for the secondary audio stream.

Examples best serve to illustrate the operation of the Encode Channel Map To Control Register:

- Consider the default output channel map from Table 4-13. This is reproduced below:

  writeENCChannelMapTo16(0,4,1,5,2,6,3,7,8,12,9,13,10,14,11,15)

  In fact, however, this is overkill. This is because only the first N_e of the sixteen entries are used for N_e-channel output by the Encode Algorithms. One of the following output channel maps will adequately specify what is required to reproduce such output. In the case of the default, 8-channel output channel map below will suffice.
- Consider the following output channel map used for 2-channel output, as to the PCE Algorithm:

  writeENCChannelMapTo2(0,1)[28]

  The effect of this *2-channel output channel map* is shown in Table 4-14 (2-Channel Output Channel Map). In this case, elements 0–1 are use for the 2-channel output, elements 2–7 are ignored, and elements 8–15 are not pertinent in PA.
- Consider the following output channel map used for 8-channel output, as to the PCE Algorithm:

  writeENCChannelMapTo8(0,4,1,5,2,6,3,7,8)

  The effect of this *8-channel output channel map* is shown in Table 4-15 (8-Channel Output Channel Map). In this case, elements 0–7 are use for 8-channel, interleaved output. As above, elements 8–15 are not pertinent in PA.

The settings that result from use of the alpha code symbols shown above, that is, writeENCChannelMapFromN_e, are appropriate for use with the SIO Output Driver provided with PA. They all require, however, an appropriate setting of the PCE Algorithm to force *N*-channel encoding, $N <= N_e$.

See also Section 4.1.5 (Audio Encode Channel Map) and Section 4.3.1 (Channel Maps).

[28] This output channel map is not actually used in PAI. It is discussed here for illustrative purposes only.

TABLE 4-14 **2-Channel Output Channel Map**

Logical Audio Channel	Bit-Stream Channel
Left	1
Rght	2

TABLE 4-15 **8-Channel Output Channel Map**

Logical Audio Channel	Bit-Stream Channel
Left	1
Rght	5
LSur	2
RSur	6
Cntr	3
Subw	7
LBak	4
RBak	8

4.2.9 Audio Output Status

The Output Buffer Audio Status Register reports on output expected from the SIO Output Device. If quiet is indicated, no output is expected because there is no audio data to output at the present time. If mute is indicated, no output is expected because output has been muted by the host. If sound but no mute is indicated, output is expected because audio data to output is available and the device is not muted by the host.

Alpha code for use with the Output Buffer Audio Status Register is given in Table 4-16 (Alpha Code for Audio Output Status).

TABLE 4-16 **Alpha Code for Audio Output Status**

Register	Alpha Code	Description
Audio Status	readOBAudio	Indicate audio output status.
	wroteOBAudioQuiet	Audio output data is not available.
	wroteOBAudioSound	Audio output data is available.
	wroteOBAudioQuietMuted	Audio output data is not available, and the output is muted under host control.
	wroteOBAudioSoundMuted	Audio output data is available, but the output is muted under host control.

4.3 Audio Miscellaneous

Various miscellaneous audio operations and concepts are discussed below.

4.3.1 Channel Maps

Both Audio Decode and Audio Encode utilize *channel maps*. A channel map takes a set of “input channels” and routes them to a set of “output channels” according to values specified in a set of control registers. This routing is not direct from input to output, but rather it occurs via a conceptual “internal” set of channels. The “input channels” and “output channels” here are inputs and outputs to and from the channel map, respectively, and they are not inputs and outputs to and from an audio stream or the system itself, as via an I/O device. To avoid confusion, the “input channels” are referred to as the *from channels*, and the “output channels” are referred to as the *to channels*. To complete this terminology, the mapping of the from channels to the internal channels is referred to as the *channel map from*, and the mapping of the internal channels to the to channels as the *channel map to*. That is, the channel map is composed of the channel map from followed by the channel map to.

The channels in each set are represented using integers 0, 1, ... , *N*-1, where *N* is the number of channels in the set. The number of channels in the input set, the internal set, and the output set for a particular channel map need not be the same. The number of channels in these sets need not be the same for different channel maps, either.

Figure 4-12 (Sample Channel Map) illustrates the general concept of a channel map.[29] This figure looks complicated, and it might be thought that implementation is difficult and consumes MIPS:

- Implementation is not difficult and does not consume MIPS. Implementation is provided as part of the Decode or Encode Algorithms in a simple manner through pointer manipulation.
- The channel map shown in Figure 4-12 is more complicated than many actual channel maps used in real systems. For example, during operation of most Decode Algorithms the channel map from is an identity, that is, straight through, and the channel map to produces PA/F-standard channel ordering.
- The channel map shown in Figure 4-12 includes unterminated routes in the channel map from. Unterminated arrows pointing down indicate from channels that are ignored and not mapped onto internal channels. Unterminated arrows pointing up indicate internal channels that are either loaded with zeros or uninitialized—both are possible and important cases that require support.
- The channel map shown in Figure 4-12 includes unterminated routes in the channel map to. Unterminated arrows pointing down indicate internal channels that are ignored and not mapped onto to channels. Unterminated arrows pointing up indicate to channels that are either loaded with zeros or not loaded—both are possible and important cases that require support.

[29] This figure does not illustrate the “bracketing” of the Decode or Encode Algorithm by the channel map. This topic is introduced further below.

It will be found to be illustrative to actually draw the channel map as shown in Figure 4-12 for various frameworks and input/output routings, both for standard and custom I/O settings. Such drawings for PA are not currently provided.

This internal set of channels is what a Decode or Encode Algorithm operates on. It can be said that the channel map *brackets* the Decode or Encode Algorithm. That is, the channel map from precedes the algorithm and the channel map to follows it. As intimated above, the actual implementation of the channel map is integrated with the algorithm that it brackets, so that the implementation cost of the channel map manifests itself as minor complexity within the algorithm but with no MIPS impact.

Current versions of the AC3, DTS, and AAC Algorithms do not provide the channel map. These are known errata.

The channel maps for Audio Decode and Audio Encode are independent and operate differently. In fact, they may be considered to operate in a manner that is *inverse* to each other. This is because the Audio Decode Channel Map routes system input channels via the decoder to the audio stream, while the Audio Encode Channel Map routes the audio stream via the encoder to system output channels. Thus, for equivalent Decode and Encode Algorithms, such as the PCM and PCE Algorithms, the Decode Channel Map To and the Encode Channel Map From are identical, and the Decode Channel Map From and the Encode Channel Map To correspond closely.

For Audio Decode in PAI and PAY:

- The from set has 2 channels for digital input, 2 channels for stereo analog input, and 8 channels for multi-channel analog input.
- The internal set has 16 channels.
- The to set has 16 logical channels but only 8 physical channels.

For Audio Encode in PAI:

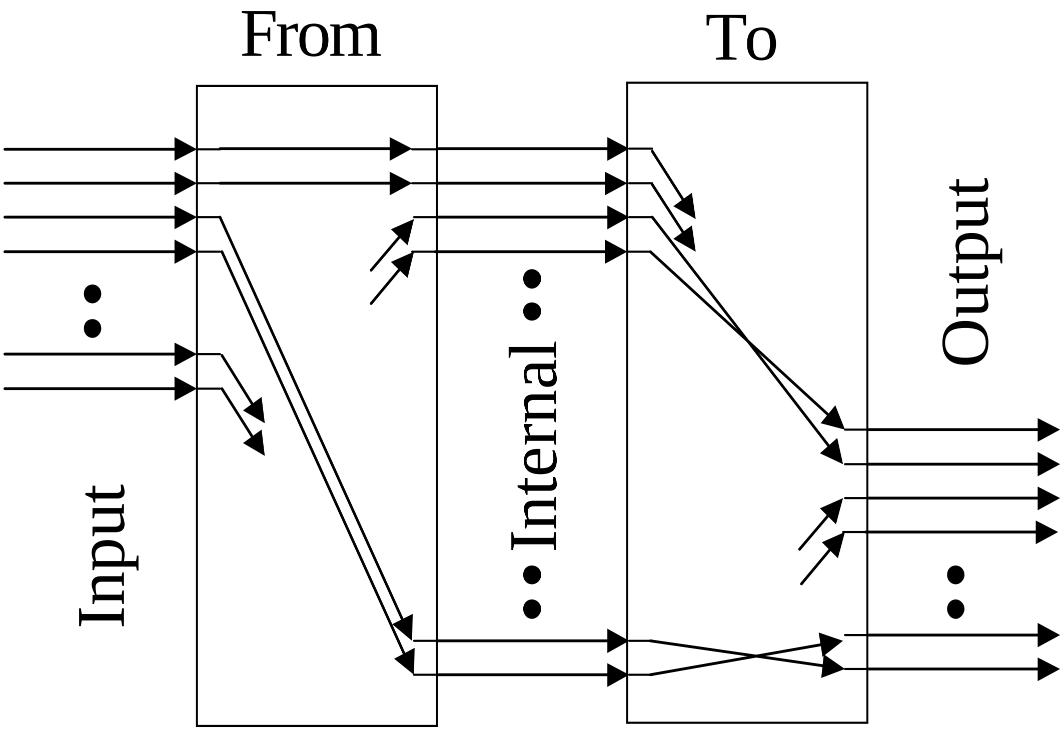
- The from set has 16 logical channels but only 8 physical channels.
- The internal set has 16 channels.
- The to set has 8 channels for multi-channel output.

For Audio Encode in PAY, which has two Encode Channel Maps for two Audio Streams:

- Each from set has 16 logical channels, but that of the Primary Audio Stream has only 8 physical channels and that of the Secondary Audio Stream has only 2 physical channels.
- The internal set has 16 channels.
- The primary to set has 8 channels for multi-channel output, and the secondary to set has 2 channels for stereo output.
- The primary to set has 8 channels for multi-channel output, and the secondary to set has 2 channels for stereo output.

These restrictions and settings are a function of the PA framework. They are intrinsic to neither the channel maps nor the algorithms involved.

FIGURE 4-12 **Sample Channel Map**



CHAPTER 5 Application Interface - Audio Stream Processing and Volume Control

This chapter, along with the previous two chapters, introduces the PA Application Programming Interface (API). The API topics covered here primarily deal with the *input*, *decode*, *audio stream processing*, *encode* and *output* components. The API topics are divided amongst these chapters as follows:

- Chapter 2 (Application Interface - Registers and Messaging)
- Chapter 3 (Application Interface - Audio Input and Decode)
- Chapter 4 (Application Interface - Audio Encode/Output)

In addition:

- Chapter 5 (Application Interface - Audio Stream Processing and Volume Control) provides a functional view of the API, conveniently grouped together under the topics of *Listening Mode*, *Speaker Configuration*, *Audio Stream Processing*, *Synchronous Rate Conversion* and *Volume Control* and covers the topic of increasing and decreasing the MIPS Load on the CPU.

5.1 System Stream Processing Functionality

As described in detail in Section 1.1 (Performance Audio Framework (PA/F) Overview), an *Audio Stream* is typically created by a Decode Algorithm, processed by a sequence of Audio Stream Processing (ASP) Algorithms, and consumed by an Encode Algorithm. The ASP Algorithms are invoked to operate on the data contained in Audio Frames as needed to process that data. This *data-driven* processing is realized as part of the Audio Stream Task.

Each Audio Stream has an associated *System Stream*. This System Stream is an Idle Task within the DSP/BIOS operating system that performs continuous operations to transform information present in one part of the Performance Audio System, and in particular within itself and within the Audio Stream with which it is associated, into information in other parts of the Performance Audio System. This “continuous operation” as part of an Idle Task creates *execution-driven* processing, where (1) all remaining processing capabilities of the device are dedicated to achieving the minimum delay in performing such processing, but (2) no demands are placed on other audio or control processing capabilities of the device at a priority other than this lowest priority.

The System Stream orchestrates a number of essential operations. It coordinates components so as to reduce the requirements on a controlling microcontroller. It can be disabled if the user so wishes, but if the System Stream is disabled, something else in the system (e.g., a microcontroller) will have to perform the disabled functions. The next sections discuss the functionality performed by the System Stream.

The Audio System Mode Control Register determines what features of the audio system are enabled. See Table 5-2 (Audio System Mode Values).

By default, all features including the three currently unused ones are enabled.

TABLE 5-1 **Alpha Code for Audio System**

Register	Alpha Code	Description
Status	readSYSStatus	Return entire SYS status structure.
Mode	readSYSMode	Return System Mode Control Register value.
	writeSYSModeDisable	Disable system operation.
	writeSYSModeEnable	Enable full system operation.
	writeSYSModeN(*N*)	Set System Mode Control Register value to enable (0x7f), enable partial (0x01–0x7e), or disable (0x00) system operation.

TABLE 5-2 **Audio System Mode Values**

Bit	Mask	Symbol	Control Feature
0	0x01	`PAF_SYS_MODE_DEC`	Decode Algorithm
1	0x02	`PAF_SYS_MODE_BM`	BM Algorithm
2	0x04	`PAF_SYS_MODE_DEM`	DEM Algorithm
3	0x08	`PAF_SYS_MODE_THX`	Reserved for THX Algorithms
4	0x10	`PAF_SYS_MODE_UNUSED4`	Unused (future features)
5	0x20	`PAF_SYS_MODE_UNUSED5`	
6	0x40	`PAF_SYS_MODE_UNUSED6`	
7	0x80	`PAF_SYS_MODE_RESERVED`	Reserved

As shown in Table 5-2 (Audio System Mode Values) the System Stream is comprised of tasks controlling:

- Listening Modes (`PAF_SYS_MODE_DEC`) - see Section 5.1.1 (Listening Mode)
- Speaker Configurations (`PAF_SYS_MODE_DEC`, `PAF_SYS_MODE_BM`) - see Section 5.1.3 (Speaker Configuration)

- Deemphasis processing (`PAF_SYS_MODE_DEM`) - see Section 5.1.5 (System CPU Load)
- Optionally THX (`PAF_SYS_MODE_THX`) - see the PA17 THX User’s Guide

5.1.1 Listening Mode

A *Listening Mode* is a collection, or set, of control register settings that produce a desired output. The effect of a Listening Mode is to convert an input stream from its current form to the desired form. It is important to understand that the Listening Mode controls how the system responds as a whole, and not how any individual component responds.

The settings of the PA control registers, especially the System Recreation Mode Control Register but also many others, determine how the audio will be generated at the output of PA and sound at the listener’s speakers. That is, the settings of these control registers determine the *Listening Mode*. If the Listening Mode calls for fewer channels than the program material supplies, *downmix* is performed automatically. Conversely, if the Listening Mode calls for more channels than the program material supplies, surround processing such as Dolby Pro Logic IIx, DTS Neo:6 2-Channel Matrix, or DTS ES Matrix 6.1 is performed automatically if appropriate. Other forms of processing, including bass management, may also be engaged automatically by PA to realize the Listening Mode that is called for.

The “Listening Mode” is a mechanism that controls many facets of the operation of PA. Alpha codes for the Listening Modes are shown in Table 5-4 (Alpha Code for Listening Modes). Default settings are shown with a shaded background. In this table, the Listening Mode Register indicates the *primary* listening mode. Listening Modes pertaining to THX are described in the PA17 THX User’s Guide.

The Standard Listening Mode (`execSTDListeningModeStandard`) is the default. Also available is the Pure Listening Mode (`execSTDListeningModePure`). They are defined as:

- Standard Listening Mode:
  - **Function**: used to restore normal, standard operation. This includes, among other things, setting the Recreation Mode to Auto.
  - **Commands used**:

    ASP Chain enabled (`writeDECASPGearControlAll`)

    SYS Recreation Mode: Auto (`writeSYSRecreationModeAuto`)

    VOL Implementation enabled and set to Internal (`writeVOLImplementation-Internal`)
- Pure Listening Mode:
  - **Function**: causes output to be rendered in a “pure” form relative to how it is received as a bit stream: no up- or downmix by the decoder, no audio stream processing, and no volume implementation as part of encoder. This includes, among other things, setting the Recreation Mode to Don’t.

- **Commands used**:

  ASP Chain disabled (`writeDECASPGearControlNil`)

  SYS Recreation Mode: Dont (`writeSYSRecreationModeDont`)

  DEC Channel Configuration Request: Unknown (`writeDECChannelConfigurationRequestUnknown`)

  VOL Implementation disabled (`writeVOLImplementationInactive`)

Certification Requirement:

> *Dolby requires down-mixing whenever the number of speakers either physically present or used to produce audio is less than the number of channels in a Dolby Digital or Dolby Pro Logic IIx program. Control of this down-mixing can be performed by PA automatically through the use of listening modes as described here and through speaker configuration as described in* Section 5.1.3 (Speaker Configuration)*. See section 4.11.4 "Listening Mode Selection" of the* Dolby Digital Licensee Information Manual, Issue 5.

Example 5-1 (Listening Mode) demonstrates some of the Listening Mode control features covered in this section.

Sub-Listening Modes

There are two types of sub-listening modes, *creation* and *recreation*. Creation involves adding new effects or synthesizing acoustic environments that are not part of the original program content. Recreation involves reproducing the full soundfield originally recorded in the source material. Each of these modes is available under any of the primary listening modes as indicated by the Listening Mode Register. At present, there are only recreation listening modes available.

As part of the recreation mode, channel configurations are requested of algorithms such as decode and surround processing so as to create the desired Recreation Mode. These algorithms respond to the request with output that best fits the request. Fourbasic kinds of Recreation Modes are provided by PA:

- **Auto:** `writeSYSRecreationModeAuto` sets the Recreation Mode to *Automatic*.

  The automatic mode derives channel and output configurations only from Speaker Control registers. Further, the Speaker Size information specified by the Speaker Control registers automatically defines the bass management redirection behavior.
- **Specific:** e.g., `writeSYSRecreationModeStereo` sets the Recreation Mode to *a specific value* such as Stereo in this case.

  The specific modes are shortcuts to some often-used Channel Configurations. The specific modes utilize the Speaker Control registers to convey the Speaker Size information, but the channel configuration is embedded in the specific mode itself. This mode is useful for selecting Mono output channel configurations, since the Auto Recreation mode does not allow for disabling the Main Left/Right channels.

- **Direct:** `writeSYSRecreationModeDirect` sets the Recreation Mode to *Direct*.

  The direct mode allows for channel and output configurations to be requested by the user "directly" using the SYS Channel Configuration Request register. The Speaker Size information specified by the Speaker Control registers is ignored in direct mode, and thus no "automatic" configuration of bass management redirection is performed. This mode is useful for selecting Phantom1 and Surround1 channel configurations, since the Speaker Control registers do not support them.
- **Dont:** `writeSYSRecreationModeDont` sets the Recreation Mode to *Dont*.

  The dont mode allows for channel and output configurations to be manipulated "directly", with no influence from the System Control registers in the System Stream task. In this mode, the SYS Channel Configuration Request and Speaker Control registers are ignored and the user has direct control over channel configuration and bass management redirection.

TABLE 5-3 **PA/F Alpha Commands Effective for Recreation Modes**

Alpha commands	Auto	Specific	Direct	Dont
writeSYSSpeakerXXX	Yes	Yes	No	No
writeSYSChannelConfigurationRequestYYY	No	No	Yes	No
writeDECChannelConfigurationRequestYYY	No	No	No	Yes
writeBMOCSelect_XXXY	No	No	Yes	Yes
writeBMOCAutoN(*N*)	Yes	Yes	Yes	Yes
writeBMOCSelectAutoN(*NN*)	No	No	Yes	Yes

The Recreation Modes are summarized in Table 5-4 (Alpha Code for Listening Modes).

The Automatic Recreation Mode (`writeSYSRecreationModeAuto`) is the default.

Channel and Output Configurations

An important intermediate mechanism through which the Listening Mode is realized are *channel* and *output configurations*. These values are derived from the Listening Mode and passed to the various algorithms of the Performance Audio Framework as a *request* for certain kinds of processing. Channel configurations are requested of algorithms such as decode and surround processing, and these algorithms respond to the request with output that best fits the request without obviating operation of other algorithms. Output configurations are requested of algorithms such as bass management, and these algorithms implement desired processing directly. The method for realizing requested configurations is discussed in Section 5.3.1 (First, Best Fit Method).

System Channel Configuration Request Select Register

The System Channel Configuration Request Select Register indicates the derived channel configuration that is used to control the decode algorithm. Since this is a select register, recall from Section 2.1.1 (Control and Status Registers) that it can sometimes be a control register, and sometimes a status register. This depends on the Recreation Mode:

- If the Recreation Mode is **automatic** or **specific**, the System Channel Configuration Request Select Register is used as a status register. It indicates the derived channel configuration that is used to control the decode algorithm.
- If the Recreation Mode is **direct**, the System Channel Configuration Request Select Register is used as a control register. It provides the channel configuration that is used to control the decode algorithm, if any is so used.
- If the Recreation Mode is **dont**, the System Channel Configuration Request select Register is unused.

Alpha code to realize the execution commands described here and for use with the registers described in this section is given in Table 5-4 (Alpha Code for Listening Modes).

TABLE 5-4 **Alpha Code for Listening Modes**

Register	Alpha Code	Description
Listening Mode	readSYSListeningMode	Return System Listening Mode Status Register value.
	execSTDListeningModeStandard	Select standard listening mode as primary.
	wroteSYSListeningModeStandard	Standard operation selected as primary.
	execSTDListeningModePure	Select pure listening mode as primary.
	wroteSYSListeningModePure	Pure operation selected as primary.
Recreation Mode	readSYSRecreationMode	Return System Recreation Mode Control Register value.
	writeSYSRecreationModeAuto[a]	Automatic mode: Channel and output configurations are derived from the Speaker Control Registers automatically.
	writeSYSRecreationModeMono	Specific mode: Mono output without subwoofer.
	writeSYSRecreationModeStereo[b]	Specific mode: Stereo output without subwoofer. Default for PAY secondary stream only.
	writeSYSRecreationMode2Stereo	Specific mode: Stereo output with subwoofer.
	writeSYSRecreationMode3Stereo	Specific mode: Front output with subwoofer.
	writeSYSRecreationModePhantom	Specific mode: Phantom output with subwoofer.
	writeSYSRecreationModeSurround	Specific mode: Surround output with subwoofer.
	writeSYSRecreationModeDirect	Direct mode: The value of the System Channel Configuration Request Select Register is used to set the channel configuration requested of the decode algorithm. No output configuration is set in the bass management algorithm.
	writeSYSRecreationModeDont	Don’t mode: No channel or output configuration is promulgated to decode and audio stream processing algorithms.
Channel Configuration Request	readSYSChannelConfigurationRequest	Return channel configuration used to control the decode algorithm.
	writeSYSChannelConfigurationRequest*X*	See Table 2-4 (Alpha Code for Channel Configuration Registers).

Register	Alpha Code	Description
Channel Configuration Request Type (see Section 5.3.1, Section 5.3.2, Section 5.3.3 and Table 5-5, Table 5-6, Table 5-7 for more information)	readSYSChannelConfigurationRequestType	Return System Channel Configuration Request Type Control Register value.
	writeSYSChannelConfigurationRequestTypeStandard	The system channel configuration request is made via the Decode Channel Configuration Request Select Register to yield a standard channel configuration request operation without override. *Downmix is standard in the Decode Algorithm.*
	writeSYSChannelConfigurationRequestTypeDecodeBypass	The system channel configuration request is made via the Decode Channel Configuration Override Select Register to yield a standard channel configuration request operation with override. *Downmix is bypassed in the Decode Algorithm.*
	writeSYSChannelConfigurationRequestTypeDecodeDirect	The system channel configuration request is made via the Decode Channel Configuration Override Select Register to yield a non-standard channel configuration request operation with override. *Downmix is non-standard in the Decode Algorithm.*

a. This is the default recreation mode for the primary audio stream.

b. This is the default recreation mode for the secondary audio stream.

5.1.2 Speaker Configuration Request Type and Decode Channel Configuration Request/Override

As is clear from above, there are multiple ways to contol the channel configuration requested from the decoder. Another factor to consider is the Decode Channel Configuration Override Select Register which controls the output of the ASPs after the decoder as explained in Section 3.2.3, "Audio Decode Channel Configuration," on page 3-60. If set to PAF_CC_UNKNOWN, the Decode Channel Configuration Request Select Register is used to control the output of the ASPs. If set to any other value, the Decode Channel Configuration Override Select Register is used to control the output of the ASPs after the decoder. The following tables explain the relation between the Speaker configuration Request Type and the decoder channel configurations.

TABLE 5-5 **SYSChannelConfigurationRequestTypeStandard**

	Dont	Auto	Direct
Decode Channel Configuration Request	Set directly by the user	Set from Speaker control registers automatically.	Set using the System Channel Configuration Request Select Register
Decode Channel Configuration Override	Set directly by the user	Always set to PAF_CC_UNKNOWN	Always set to PAF_CC_UNKNOWN

TABLE 5-6 **SYSChannelConfigurationRequestTypeDecodeDirect**

	Dont	Auto	Direct
Decode Channel Configuration Request	Set directly by the user	Set directly by the user	Set directly by the user
Decode Channel Configuration Override	Set directly by the user	Set from Speaker control registers automatically.	Set using the System Channel Configuration Request Select Register

TABLE 5-7 **SYSChannelConfigurationRequestTypeDecodeBypass**

	Dont	Auto	Direct
Decode Channel Configuration Request	Set directly by the user	Always set to PAF_CC_UNKNOWN	Always set to PAF_CC_UNKNOWN
Decode Channel Configuration Override	Set directly by the user	Set from Speaker control registers automatically.	Set using the System Channel Configuration Request Select Register

EXAMPLE 5-1 **Listening Mode**

This example demonstrates the listening mode control features of PA.

Return PA to its default operating mode Play a Dolby Digital 5.1 bit stream, one with program material on all six channels.

Listen, and output should be audible on all eight channels (Dolby Digital EX is enabled by default and produces the back channels). View the Listening Format Status Register using alpha code `readENCListeningFormat`. The response `0x1a07,0x0000` indicates 3/4.1 reproduction, if interpreted as described in Section 5.4.4 (Audio Encode Listening Format), especially Table 5-18 (Encode Listening Format Status Register -- Bitmap).

Now, change the listening mode to mimic reproduction in a stereo-plus-subwoofer system using alpha code `writeSYSRecreationMode2Stereo`. Again view the Listening Format Status Register using alpha code `readENCListeningFormat`. The listening format will be `0x1003,0x0000`, indicating 2/0.1 reproduction. Output will be audible on only the left, right, and subwoofer channels.

5.1.3 Speaker Configuration

The Speaker Control Registers are used to specify the presence, absence, and size of speakers. These registers are used in the generation of channel and output configurations in automatic and specific recreation listening modes. They are not utilized when direct or don't recreation listening modes are in use.

Some examples of this are:

- Selecting a large center speaker will not guarantee that full range output will be routed to the center channel output. Rather, selecting a large center speaker will cause an output configuration for bass management to be chosen that can best utilize the fact that the center speaker can reproduce audio over the full frequency range.
- Selecting no subwoofer speaker will not guarantee that no signal will be routed to the subwoofer output. Rather, some instantiations of the Performance Audio Framework may cause generation of subwoofer output even though this register indicates that there is no associated speaker. The Bass Management component typically operates to redirect to available speakers.
- Selecting back speaker(s) present and surround speakers absent (e.g., "writeSYSSpeakerBackSmall2,writeSYSSpeakerSurrNone") does *not* correspond to a PA-supported speaker configuration. In most cases, this will result in output being sent to the surround speakers and no output being sent to the back speakers.

Alpha code for use with the control and status registers described in this section are given in Table 5-8 (Alpha Code for Speaker Configuration). These registers do not affect the spectral content of the output directly, but rather their values are interpreted and used to select channel and output configurations.

For example, from Table 5-8:

- Send alpha code `writeSYSSpeakerSubwNone` to indicate the absence of the subwoofer speaker.
- Send alpha code `writeSYSSpeakerSurrLarge2` to indicate two large surround speakers.

TABLE 5-8 **Alpha Code for Speaker Configuration**

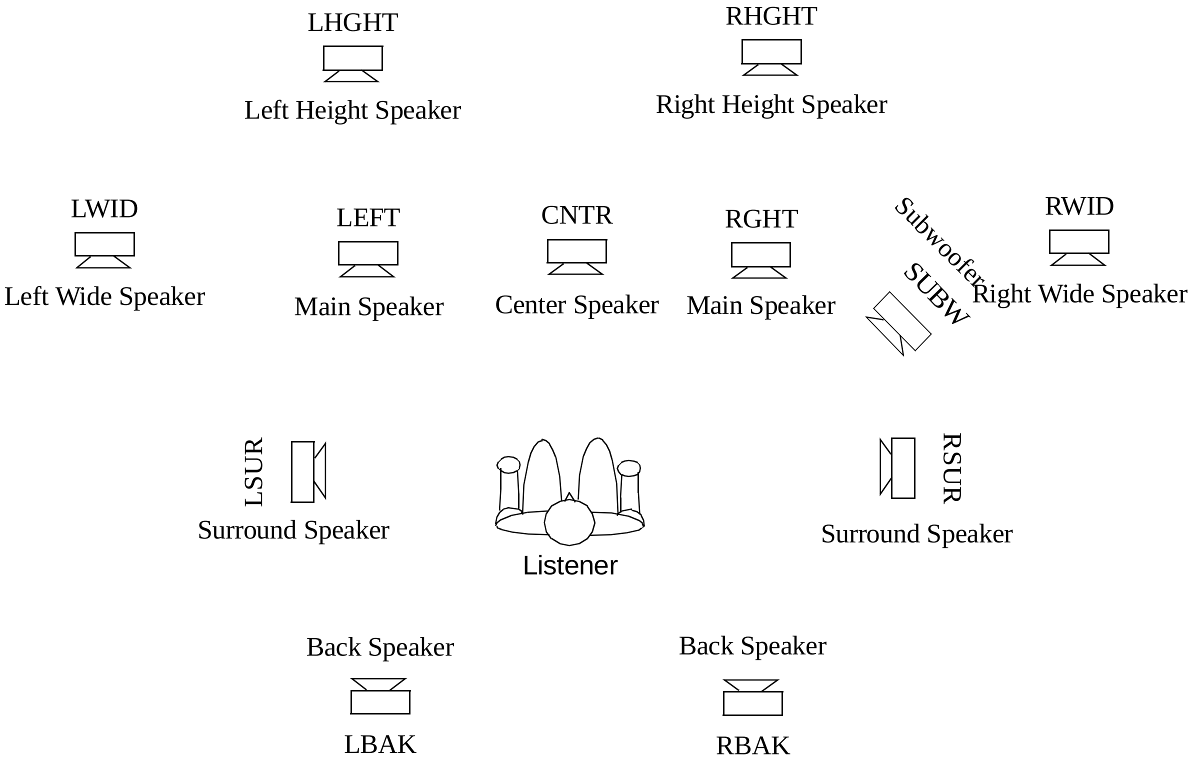
Register	Alpha Code	Description
Speaker Group	readSYSSpeaker*Group*	Return System Speaker Control Register value for Main, Cntr, Surr, Back, and Subw groups.
Main Speaker	writeSYSSpeakerMainSmall2	Small speakers for the left and right channels.
	writeSYSSpeakerMainLarge2	Large speakers for the left and right channels.
Center Speaker	writeSYSSpeakerCntrNone	No speakers.
	writeSYSSpeakerCntrSmall1	One small speaker.
	writeSYSSpeakerCntrLarge1	One large speaker.
Surround Speakers	writeSYSSpeakerSurrNone	No speakers.
	writeSYSSpeakerSurrSmall2	Two small speakers.
	writeSYSSpeakerSurrLarge2	Two large speakers.

Register	Alpha Code	Description
Back Speakers	writeSYSSpeakerBackNone	No speakers.
	writeSYSSpeakerBackSmall1	One small speaker.
	writeSYSSpeakerBackLarge1	One large speaker.
	writeSYSSpeakerBackSmall2	Two small speakers.
	writeSYSSpeakerBackLarge2	Two large speakers.
Height Speaker	writeSYSSpeakerWideNone	No speakers.
	writeSYSSpeakerWideSmall2	Two small speaker.s
	writeSYSSpeakerWideLarge2	Two large speakers.
Wide Speakers	writeSYSSpeakerWideNone	No speakers.
	writeSYSSpeakerWideSmall2	Two small speakers.
	writeSYSSpeakerWideLarge2	Two large speakers.
Subwoofer Speaker	writeSYSSpeakerSubwNone	No speakers.
	writeSYSSpeakerSubwBass1	One bass speaker.

The location of the speakers corresponding to the registers in Table 5-8 are shown in Figure 5-1 (Speaker Location for Multi-channel Output).

Example 5-2 (Speaker Configuration) provides a detailed demonstration of many of the speaker configuration features of PA.

FIGURE 5-1 **Speaker Location for Multi-channel Output**



Certification Requirement:

> *Dolby requires that the product have knowledge of the number and types of speakers attached to the product in order to deliver program material appropriately. Several interfaces to control speaker options are discussed in section 4.11.1 "Speaker set-up" of the* Dolby Digital Licensee Information Manual, Issue 5. *These interfaces can be implemented using the features of PA described above.*

EXAMPLE 5-2 **Speaker Configuration**

This example demonstrates the speaker configuration features of PA.

Return PA to its default operating mode Play a Dolby Digital 5.1 bit stream, one with pleasing program material on all six channels.

In the default operating mode, output on left, center, right, left surround, right surround, right back, left back and subwoofer speakers is selected. Output should be audible on the selected channels. View the Listening Format Status Register using alpha code `readENCListeningFormat`. The response `0x1a07` indicates 3/4.1 reproduction, if interpreted as described in Section 5.4.4 (Audio Encode Listening Format), especially Table 5-18 (Encode Listening Format Status Register -- Bitmap).

Now, change the speaker configuration several times as described below, listening to the output and interpreting the Listening Format Status Register as above after each selection:

- `writeSYSSpeakerCntrNone -` No center speaker is selected, and the center output will become silent. The listening format will be `0x1a03`, indicating 2/4.1 reproduction.
- `writeSYSSpeakerBackNone -` No back speakers are selected, and the back outputs will become silent. The listening format will be 0x1203 indicating 2/2.1 reproduction.
- `writeSYSSpeakerSurrNone -` No surround speakers are selected, and the surround outputs will become silent. The listening format will be`0x1003`, indicating 2/0.1 reproduction.
- `writeSYSSpeakerCntrSmall1 -` One small center speaker is selected, and the center output will become audible. The listening format will be `0x1007`, indicating 3/0.1 reproduction.
- Alpha code sequence `writeSYSSpeakerMainLarge2, writeSYSSpeakerCntrNone, writeSYSSpeakerSubwNone -` Two large left and right speakers, no center speaker, and no subwoofer are selected. The listening format will be `0x0003`, indicating 2/0.0 reproduction, and the left and right speakers only will produce audible output.
- Alpha code sequence `writeSYSSpeakerMainSmall2, writeSYSSpeakerCntrSmall1, writeSYSSpeakerSurrSmall2, writeSYSSpeakerBackSmall2, writeSYSSpeakerSubwBass1` - Returns the system to its original speaker configuration settings. The listening format will be restored to`0x1a07`, indicating 3/4.1 reproduction, and all speakers will again produce audible output.

5.1.4 De-emphasis Control

The De-emphasis (DEM) Algorithm Control Register manipulation enabled via System Mode Control Register bit 2 as shown in Table 5-2 (Audio System Mode Values). The System Stream provides automatic activation of de-emphasis occurs whenever it is detected that the input signal has been pre-emphasized.

5.1.5 System CPU Load

The system stream measures the ‘instantaneous’ CPU load value over intervals of approximately one half-second. From these successive ‘instantaneous’ loads, a ‘peak’ load is accumulated; the ‘peak’ represents the maximum ‘instantaneous’ load since the system was restarted or the peak was most-recently cleared. Alpha commands are provided, as shown in Table 5-9 (Alpha Code for System CPU Load), to read the ‘instantaneous’ and ‘peak’ loads, and to clear the ‘peak’ load reading.

The load readings provided by the system stream and the load readings provided by the CCS load graph may differ slightly, but the values returned by the system stream are more accurate.

When the CPU is loaded more than 100% (so that real time operation is broken), the values returned by following alpha codes will be incorrect, due to load-shedding.

TABLE 5-9 **Alpha Code for System CPU Load**

Register	Alpha Code	Description
CPU load	readSYSCpuLoad	Return the integral part of the ‘instantaneous’ CPU load in 8-bit Q0 format. For a CPU load of 23.56%, returns 0x17 (23).
	readSYSCpuLoadQ8	Return the ‘instantaneous’ CPU load in 16-bit Q8 format, rounded to integral percentage. For a CPU load of 23.00%, returns 0x1700.
Peak CPU load	readSYSPeakCpuLoad	Return the integral part of the ‘peak’ CPU load in 8-bit Q0 format. For a peak CPU load of 23.56%, returns 0x17 (23).
	readSYSPeakCpuLoadQ8	Return the ‘peak’ CPU load in 16-bit Q8 format, rounded to integral percentage. For a peak CPU load of 23.00%, returns 0x1700.
	writeSYSPeakCpuLoadClear	Set the ‘peak’ CPU load to zero, restarting peak load accumulation. The peak load reading of 0 will be overwritten when the system stream next measures the ‘instantaneous’ load. Successive ‘instantaneous’ load readings will replace the ‘peak’ load if larger.

5.2 Basic Audio Stream Processing

A brief description of the basic operation of Audio Stream Processing with PA is given here.

The location of the Audio Stream Processing Chains (just one chain for PAI) in the Performance Audio Stream is shown in the following figures for each topology:

- Figure 5-2 (Location of the ASP Chain for PAI)
- Figure 5-3 (Location of the ASP Chains for PAY)
- Figure 5-4 (Location of the ASP Chains for PAD)
- Figure 5-5 (Location of the ASP Chains for PAH)
- Figure 5-6 (Location of the ASP Chains for PAZ)

As can be seen, Audio Stream Processing takes place between audio decode and audio encode.

FIGURE 5-2 **Location of the ASP Chain for PAI**

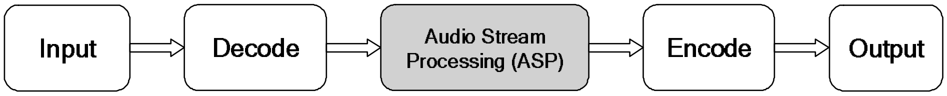


FIGURE 5-3 **Location of the ASP Chains for PAY**

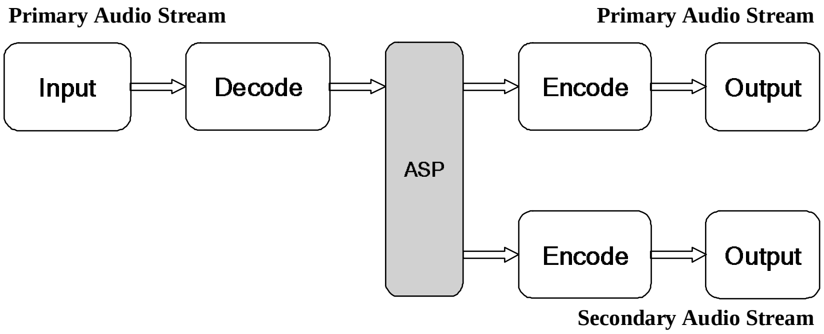


FIGURE 5-4 **Location of the ASP Chains for PAD**

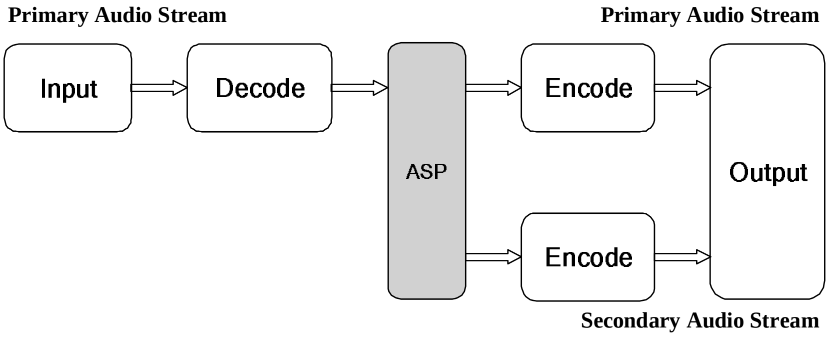


FIGURE 5-5 **Location of the ASP Chains for PAH**

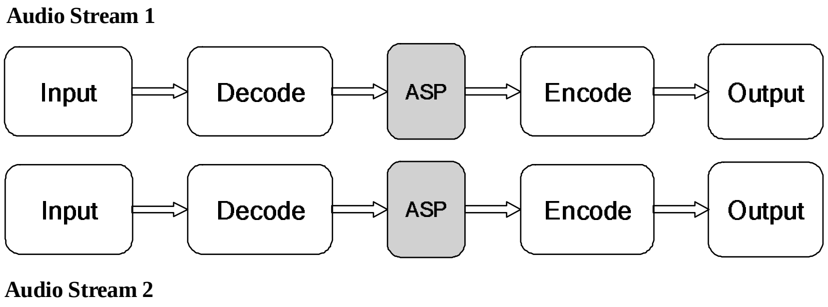
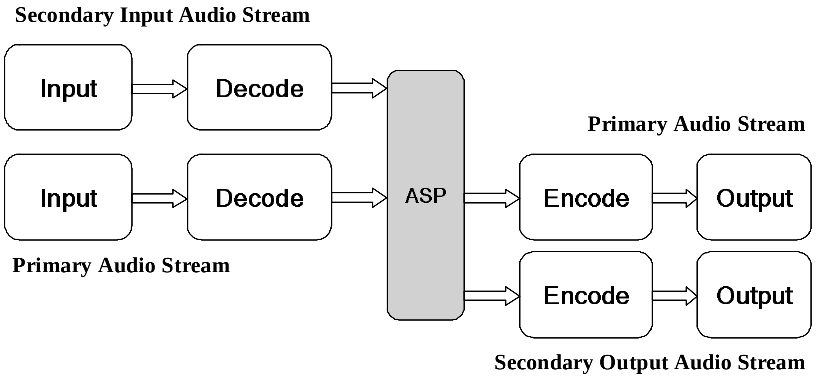


FIGURE 5-6 **Location of the ASP Chains for PAZ**



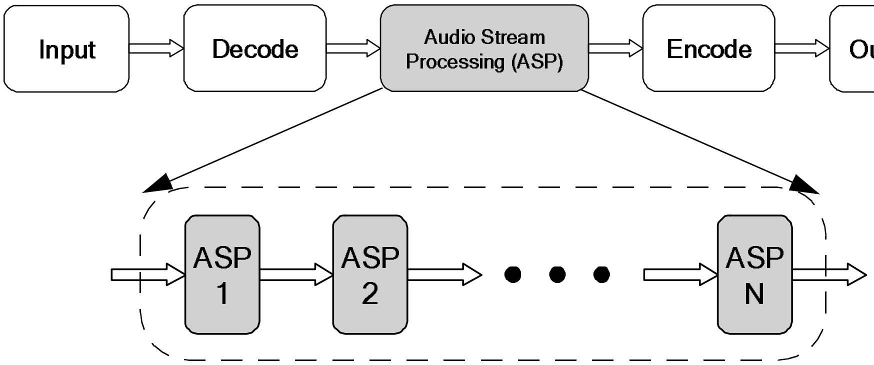
5.2.1 Audio Stream Processing Chains

5.2.1.1 PAI

As shown in Figure 5-7 (Audio Stream Processing Chain for PAI), below, the *Audio Stream Processing Chain* implements a cascade of Audio Stream Processing Algorithms, one after each other. Each ASP Algorithm is labeled *ASP* in the figure.

In PAI, there is a single ASP Chain that processes the output of the single, active Decode Algorithm and generates input for the single, active Encode Algorithm.

FIGURE 5-7 **Audio Stream Processing Chain for PAI**



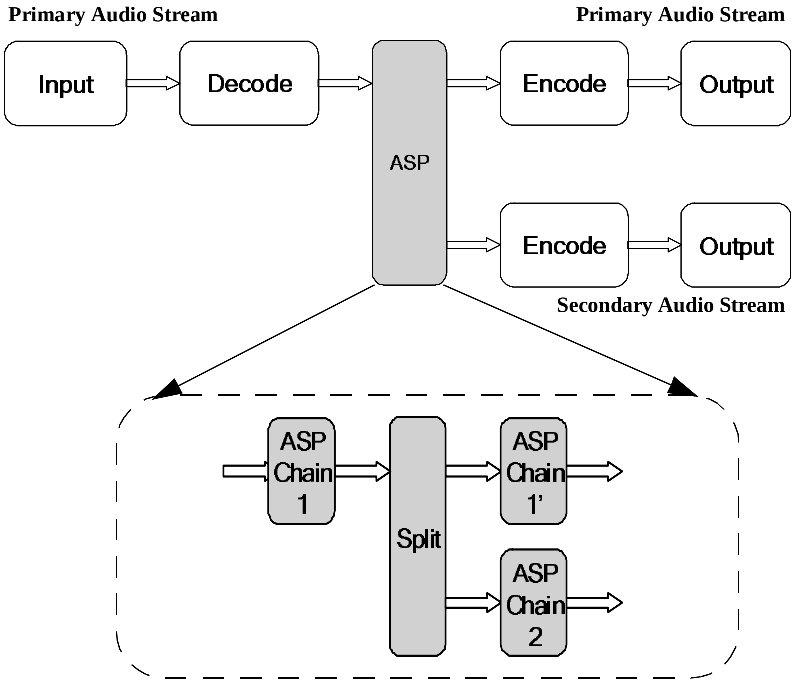
The order of the ASP Algorithms is important, and is as per the order listed above.

5.2.1.2 PAY

As shown in Figure 5-8 (Audio Stream Processing Chains for PAY) below, the *Audio Stream Processing Chains* implements a cascade of Audio Stream Processing Algorithms, one after each other. Each ASP Algorithm is labeled as *ASP* in the figure.

In PAY, there are two, split ASP Chains which process the output of the single, active Decode Algorithm. Using the notation from Figure 5-8, ASP *Chain 1 and 1'* represent the *Primary ASP Chain* and ASP *Chain 2* represent the *Secondary ASP Chain*. The ASS Algorithm passes its input as a primary output in the Primary ASP Chain, but it also generates from its input a secondary output for use as the input to the Secondary ASP Chain. The two, split ASP Chains generate the input for the two, dual, active Encode Algorithms.

FIGURE 5-8 **Audio Stream Processing Chains for PAY**



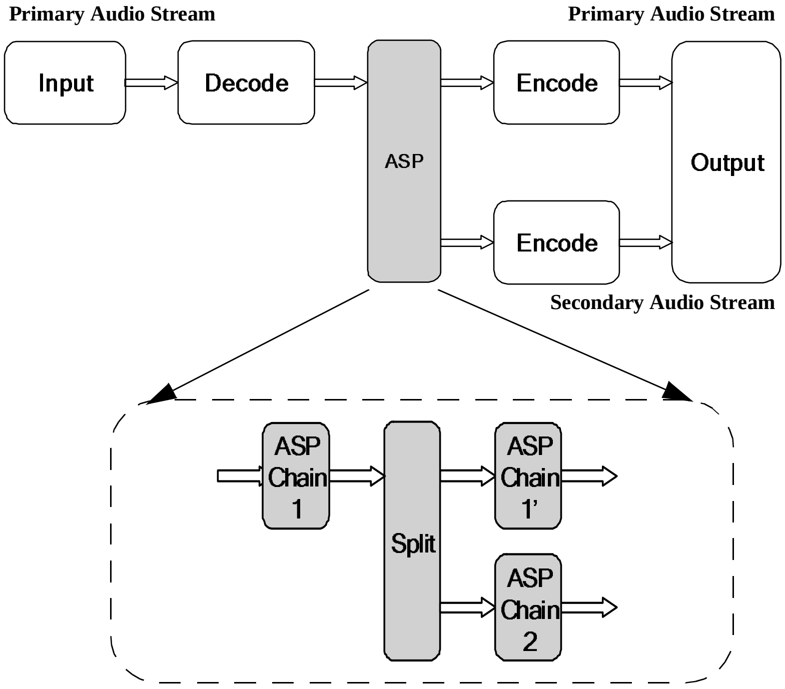
In PAY, the Primary ASP Chain consists of the following Audio Stream Processing Algorithms:

5.2.1.3 PAD

As shown in Figure 5-8 (Audio Stream Processing Chains for PAY) below, the *Audio Stream Processing Chains* implements a cascade of Audio Stream Processing Algorithms, one after each other. Each ASP Algorithm is labeled as *ASP* in the figure.

In PAD, like PAY, there are two, split ASP Chains which process the output of the single, active Decode Algorithm.

FIGURE 5-9 **Audio Stream Processing Chains for PAD**



In PAD, the Primary ASP Chain consists of the following Audio Stream Processing Algorithms:

5.2.1.4 ASP Algorithm Control and Status Registers

The operational mode of an ASP algorithm is determined in part by its control registers. Such control is typically not performed on an on-going basis, but rather only when change in operation is desired.

It is often the case that an ASP Algorithm will use a particular control register to set an *application level*. The idea is that if the audio stream meets certain criteria set forth by the algorithm when it reaches that algorithm in the ASP chain, then the ASP will activate itself and process the audio stream. If these criteria are not met, then the ASP will not process the audio stream. The application level controls what criteria must be met. Usually, the criteria range from strict to simple. On-off controls are always provided at the high and low ends of the applicability scale.

5.2.1.5 Audio Stream Control and Status Registers

The operational mode of an algorithm is determined in part by the register values associated with the audio stream on which it operates. These register values describe various features of the audio stream and provide control over various aspects of the algorithms.

5.3 Audio Stream Processing Fit Method

Not all ASP Algorithms process the audio data at all times. Thus, some mechanism must be used by each ASP Algorithm in the chain to decide whether or not to use its capabilities to process the audio data. The method used in Performance Audio is described as the *first, best fit method*.

5.3.1 First, Best Fit Method

The *first*, *best fit method* is the mechanism used by each ASP Algorithm in the chain to decide whether or not to use its capabilities to process the audio data.

A *first fit* gives priority naturally. ASP Algorithms operate within the Audio Stream Processing Chain, one after another following the decode and downmix portions of the Decode Algorithm itself. An ASP Algorithm earlier in the ASP Chain always has the option to operate on the audio stream to convert it to the requested form before a later ASP Algorithm. An example provides the best explanation for this process:

- The implementation of Dolby Pro Logic via the Pro Logic IIx (PL2x) Algorithm precedes the implementation of DTS Neo:6 2-Channel Matrix via the NEO Algorithm in the ASP Chain. Thus, all things being equal, Dolby Pro Logic IIx will be used rather than DTS Neo:6 2-Channel Matrix for surround processing to convert stereo audio signals to multi-channel audio signals.

A *best fit* is always preferred, thus things are not always equal. Certain Decode, Audio Stream Processing, or Encode Algorithms are better suited to process audio input in certain situations. Several examples provide the best explanation for this term:

- DTS ES Neo:6 2-Channel Matrix typically provides a better mechanism than Dolby Pro Logic IIx for surround processing of DTS 2-channel audio source material. Thus, Dolby Pro Logic IIx is not typically used to process DTS input, even though it precedes DTS ES Neo:6 in the ASP chain.
- Bass Management typically provides a better mechanism than a Decode Algorithm for deciding how to route low-frequency audio signals among the various channels.[30] Thus, a Decode Algorithm will typically output low-frequency audio portions of all signals as it is delivered in the program rather than providing any special low-frequency processing.

It is quite important to note the liberal use of the term “typically” in the examples above. All such decisions ultimately need to be under the control of the host. The *application level*, as defined in Section 5.2.1.4 (ASP Algorithm Control and Status Registers), is usually provided and used for exactly this purpose.

Note that the generation of the fit is also part of the operation of the Decode and Encode Algorithms. In particular, the *decode* and *downmix* portions of all Decode Algorithms operate in concert to provide the fit even before the audio stream is released from its

[30] The AC3 Algorithm is notorious in this respect, where LFE information is simply thrown away unless explicitly requested as subwoofer output.

clutches to the Audio Stream Processing Chain. And, the Encode Algorithm provides final fit, especially including volume manipulations, after the audio stream is released from the same clutches.

It is not always possible for a single algorithm, a pair of algorithms, or even the whole collection of algorithms that make up the decode-Audio Stream Processing-encode sequence, to convert the audio stream to its desired form. For example, if Lt,Rt output is requested, the output will not be Lt,Rt unless the input is Lt,Rt. This is due to the lack of a Lt, Rt encoder in the system.

In such cases, the output simply does not meet the request. It is the responsibility of the host here to monitor status registers that indicate what the channel configuration of the output actually is and take any necessary corrective action.

5.3.2 Fit Methods for Surround Processing

Surround processing[31] presents a unique challenge as regards the fit method. Surround processing ASPs must operate within the bounds of the first, best fit method, but these bounds must be extended to allow the forms of surround processing of interest to the user.

- All surround processing is grouped together in the ASP Chain. It follows “input processing” such as de-emphasis but precedes processing such as bass management.
- Custom surround processing comes first. It is disabled by default.
- Dolby surround processing comes next. It is enabled but set to ignore DTS programs by default.
- DTS surround processing comes next. It is enabled but set to ignore Dolby programs by default.
- Other surround processing comes next They are enabled by default.

This operation of ASP-based surround processing follows standard processing by the Decode Algorithm, including both downmix and surround processing as applicable.

The net effect of the standard fit method for surround processing is that by default, Dolby surround processing is engaged for Dolby and PCM programs and DTS surround processing is engaged for DTS programs. When custom surround processing is enabled it will engage as appropriate. When wishing to engage other surround processing, the settings for Dolby and DTS surround processing must be altered to cause program material of interest, such as PCM, to not be engaged by those algorithms.

The alpha code symbol `readENCSampleProcess0` will return information as to what surround processing is active at any given time. Table 5-10 (readENCSampleProcess Response Symbol Values) describes how to interpret the response to this alpha code.

[31] Surround processing is termed *matrix decoding* or *matrix processing* in other literature.

TABLE 5-10 **readENCSampleProcess Response Symbol Values**

Byte(s)	Bit(s)	Name	Description
0	0	PL	Dolby Pro Logic is active (applies only to Dolby Pro Logic IIx operating on stereo input).
	1	NEO	NEO is active.
	2	ASA	ASA is active.
	3	DEX	Dolby Surround EX is active (applies only to DEX mode of Dolby Pro Logic IIx, not 6 Channel Music, 7 Channel Movie, or 7 Channel Music modes).
	4	SURRBASS	If set, bass from surround channels should not be redirected to the subwoofer.
	5	BACKBASS	If set, bass from back channels should not be redirected to the subwoofer.
	6	MONOBACK	If set, the back channels are mono (correlated even if they are split and routed to two channels).
	7	SURRPROC	If set, an algorithm producing surround channel(s) is active.
1	0	BACKPROC	If set, an algorithm producing back channel(s) is active.
	1	PL2X	Dolby Pro Logic IIx is active (applies to all uses of Dolby Pro Logic IIx).
	2	RVB	Room Simulator, Number 1 is active.
	3	MTX	Matrix is active.
	4	GEQ	Graphic Equalizer is active.
	5	SRC	Synchronous Sample Rate conversion is active.
	6	DMX	Downmix is active.
	7	DEM	Deemphasis is active.
2	0–7		Reserved for OEM applications.
3	0–7		Reserved for custom applications.

5.3.2.1 Standard Fit Method for Surround Processing

The standard fit method for surround processing, as described in Section 5.3.2 (Fit Methods for Surround Processing), is used by default in PA. To achieve this, the following system register settings are utilized:

```
alpha writeSYSChannelConfigurationRequestTypeStandard
alpha writeSYSSpeakerX
alpha execSTDListeningModeStandard,...
```

When custom surround processing ASP Algorithms are added, they should be added in a manner which preserves the operation of the standard fit method for surround processing.

5.3.2.2 Decode-Bypass Fit Method for Surround Processing

The *decode-bypass fit method for surround processing* operates as per the standard fit method, except that the Decode Algorithm provides neither downmix nor surround processing. Instead, the audio stream output by the decoder and input to the custom surround-processor is expected to be processed by the ASP Algorithms to achieve the requested channel configuration.

The net effect of the decode-bypass fit method for surround processing is similar to that of the standard fit method. In the case of Custom ASP Algorithms, this fit method allows signal processing before, rather than after, downmix and decoder-based surround processing. As noted below, it is true that in this case the Custom ASP Algorithm itself must supply this downmix and decoder-based surround processing, if desired, rather than relying on the Decode Algorithm to supply it.

The decode-bypass fit method for surround processing uses register settings such as the following:

```
alpha writeSYSChannelConfigurationRequestTypeDecodeBypass
alpha writeSYSSpeakerX
alpha execSTDListeningModeStandard,...,writeXXXEnable
```

When custom surround processing ASP Algorithms such as *XXX* are added, they should be added in a manner which preserves the operation of the decode-bypass fit method for surround processing to the extent that the system will be actually used.

5.3.2.3 Decode-Direct Fit Method for Surround Processing

The *decode-direct fit method for surround processing* operates as per the decode-bypass fit method except that the downmix and surround processing implemented by the Decode Algorithm are controlled directly using a channel configuration request. When using the decode-direct fit method, the downmix is done in the decoder.

The decode-direct fit method for surround processing uses register settings such as the following:

```
alpha writeSYSChannelConfigurationRequestTypeDecodeDirect
alpha writeDECChannelConfigurationRequestX
alpha execSTDListeningModeStandard,...,writeXXXEnable
```

When custom surround processing ASP Algorithms such as *XXX* are added, they should be added in a manner which preserves the operation of the decode-direct fit method for surround processing to the extent that the system will be actually used.

5.3.2.4 Manual Fit Method for Surround Processing

The *manual fit method for surround processing* operates as per the other fit methods above, except that the Channel Configuration Request Select Registers are not determined indirectly by the system but rather directly by the user, and thus operate as control registers.

The manual fit method for surround processing uses register settings such as the following:

```
alpha execSTDListeningModeStandard,writeSYSRecreationModeDont
alpha writeDECChannelConfigurationRequestX
alpha writeDECChannelConfigurationOverrideY
```

When custom surround processing ASP Algorithms are added, they should be added in a manner which preserves the operation of the manual fit method for surround processing to the extent that the system will actually be used.

Note: ***Downmix Must Be Performed***

If a Custom Surround Processing ASP Algorithm is used in conjunction with a fit method that bypasses or redirects downmix capability provided by the Decode Algorithm, equivalent downmix capability must be provided by that Custom Surround Processing ASP Algorithm.

PAY : Dealing with the implications of this caveat as regards the Secondary audio stream are left as an exercise for the reader.

5.3.3 Must Fit Method

A fit is required as part of Encode Processing. While Encode Processing participates as a component in the operation of the *first, best fit method,* its special location as the final link in the decode-audio stream-encode processing chain means that it is required to produce a fit *to some extent*.

The final channel usage of the Audio Stream is described by the Encode Channel Configuration Stream. The required channel usage of the encoded audio data is described by the Encode Channel Configuration Request. The job of the Encode Algorithm is to convert the Audio Stream from its current form, as given by the Encode Channel Configuration Stream, into the required form, as given by the Encode Channel Configuration Request.

If the current form is already the required form, then no additional processing other than encoding of the Audio Stream is required. If the current form is not the required form, then additional processing other than encoding of the Audio Stream is performed. This results in the *must fit* method for Encode Processing, and is summarized as follows:

- If channels required in the encoding are not present, they are supplied as zero-valued data.
- If channels not required in the encoding are present, they are ignored.

An example for each of the above situations provides the best explanation for this process:

- PCM Encoding of 7.1-channel output requires a subwoofer channel as the eight channel. If none is provided, zero-valued data is output on the eight channel.
- PCM Encoding of 2-channel output allows only two output channels. If subwoofer channel output is provided it is ignored.

As described in Section 5.3 (Audio Stream Processing Fit Method), the generation of a *first, best fit* is part of the operation of the Encode Algorithm in that it provides the final fit, especially including volume manipulations, after the Decode and Audio Stream Processing is complete.

5.4 Audio Operation Coordination

Audio operation coordination describes the interaction between control operations. In particular, additional intervention on the part of the host is required in some cases because the effects of certain control operations are *deferred* and may never be realized otherwise.

Alpha code for use with the control and status registers described in this section is given in Table 5-11 (Alpha Code for Audio Decode Audio Operation Coordination).

TABLE 5-11 **Alpha Code for Audio Decode Audio Operation Coordination**

Register	Alpha Code	Description
Command Action	readDECCommand	Return Decode Command Register value.
	writeDECCommandN(N)	Send command number (selected as the offset value from one of the available DECCommand choices).
	writeDECCommandNone	Null operation.[a]
	writeDECCommandAbort	Abort Decode Processing.[b]
	writeDECCommandRestart	Restart Decode Processing.
	wroteDECCommandNoned	Null operation.
	wroteDECCommandAborted	Aborted Decode Processing.
	wroteDECCommandRestarted	Restarted Decode Processing.

[a] This command is a NULL command and should not be normally used.

[b] This command immediately exits the decoder state machine without proper cleanup and should not be used except possibly for debugging purposes.

5.4.1 Understanding Control Operation Deferral

There are instances when a control operation requested by the host does not take effect unless further action is taken by the host. These instances are called *deferred control operations*.

The effect of many control operations seems to be felt immediately. This is almost never actually the case, however. Some kind of delay always occurs between the instant when the control operation is transmitted by the host and when its effect is felt at the listener's ears. At a minimum, this delay in the *resolution* of the control operation is a function of the communications medium and protocol and the audio transmission path. At a maximum, this can be forever if care is not taken with the request.

The important distinction here is whether, under normal operating conditions during which output is audible, the desired effect will naturally, eventually occur in its proper time, or whether some kind of additional action is required by the generator of the control operation to cause the desired effect to be realized:

- The resolution of *non-deferred control operations* may be delayed, but this delay is always for a definite period of time under normal operating conditions during which output is audible. The time period through which the effects are delayed for a non-deferred control operation may be until the next block boundary in the output audio data stream, until the next frame boundary in the input bit stream, or after the expiration of a specific time interval. Other time periods are possible. None of these time periods are universal, but the first is typical of most control operations.

  The point here is that non-deferred control operations require no other action on the part of the host to cause realization of the intended effect, that is, to resolve the control operation, at least under normal system operating conditions.
- The resolution of *deferred control operations*, on the other hand, may be delayed for an indefinite period of time under normal operating conditions during which output is audible. That is, deferred control operations may not be resolved. Events may occur that cause an effect to be felt, that is, the control operation to be resolved, but on the other hand such events may not occur.

  The point here is that deferred control operations require other action on the part of the host to guarantee realization of the intended effect, that is, to resolve the control operation, at least under normal system operating conditions.

No further discussion is required as regards non-deferred control operations. Deferred control operations are discussed further both *in situ* and in the next section.

5.4.2 Identifying Control Operation Deferral

Control operation deferral occurs in the following situations. Techniques for resolution are suggested:

- The Decode Source Select Control Register may be used to select the PCM direct form of source decoding. This causes an encoded bit stream to be decoded as PCM, which sounds like noise. In this case, changing the Decode Source Select Control Register back to select the automatic form will not cause the bit stream to be decoded as other than PCM. This is because decoding of this bit stream as PCM is considered a legal interpretation of the input bit stream.

  The control operation deferral will be resolved if the input bit stream is restarted.[32]

  To overcome the control operation deferral, Decode Processing should be reset as described in Section 5.4.3 (Overcoming Control Operation Deferral) below.[33]
- If the Decode ASP Gear Control Register[34] is changed, its effect is deferred.

  To overcome this control operation deferral, Decode Processing should be reset as described in Section 5.4.3 (Overcoming Control Operation Deferral) below.

[32] A bug in the current implementation of PA prevents resolution in this manner.

[33] A bug in the current implementation of PA prevents resolution in this manner. Rather, what is required is to first select the bit-stream form and then the automatic form of source decoding in the first place.

[34] The value in this register determines which of the Audio Stream Processing Algorithms are active, and which are disabled.

- If the sample rate is manually controlled, the effect of any change to the associated control register will be deferred.

  To overcome this control operation deferral, Decode Processing should be reset as described in Section 5.4.3 (Overcoming Control Operation Deferral) below.

Control operation deferral may occur in other situations not listed above.

5.4.3 Overcoming Control Operation Deferral

Decode Processing can be reset as shown in Table 5-11 (Alpha Code for Audio Decode Audio Operation Coordination) to resolve most forms of control operation deferral. Note that this reset operation will cause a momentary mute in analog audio output.

5.5 Audio Stream Processing Components

5.5.1 Audio Stream Processing Reset

The term *reset* or *restart* is used to describe the process by which decoding of an input bit stream is initialized or reinitialized. Audio stream processing reset can be caused in one of two ways:

1. If the source transmission is interrupted, as via a button push on the player or via a cable disconnect, audio stream processing may reset. It will reset during decoding of encoded bit streams. It will not necessarily reset during decoding of unencoded (PCM) bit streams.
2. The alpha code `writeDECCommandRestart` will cause audio stream processing reset.

5.5.2 Audio Stream Processing Gear

The *gear* provides a basic trade-off between MIPS and functionality, like that of a bicycle or automobile. The alpha codes for the Decode ASP Gear Control Register in Table 5-12 (Alpha Code for Audio Stream Processing Control and Status) specify which of the ASP Algorithms are active, and which are disabled. The default setting of this control register provides full Audio Stream Processing capabilities. Other settings can be used to disable all or a portion of the full Audio Stream Processing capabilities.

If the Decode ASP Gear Control Register is changed, its effect is deferred until Decode Processing is reset. See Section 5.4.2 (Identifying Control Operation Deferral).

The Decode ASP Gear Status Register shows the effective Audio Stream Processing capabilities that are in force at any time.

TABLE 5-12 **Alpha Code for Audio Stream Processing Control and Status**

Register	Alpha Code	Description
Gear	readDECASPGearStatus	Return Decode ASP Gear Status Register value.
	readDECASPGearControl	Return Decode ASP Gear Control Register value.
	writeDECASPGearControlAll	Shift to all Audio Stream Processing.
	writeDECASPGearControlNil	Shift to no Audio Stream Processing.
	writeDECASPGearControlStd	Shift to standard Audio Stream Processing only.[a]
	writeDECASPGearControlCus	Shift to custom Audio Stream Processing only.[b]
	writeDECASPGearControlN(*N*)	Shift to other Audio Stream Processing.

[a] Alternate, OEM, and Custom Audio Stream Processing Algorithms are disabled. Standard Audio Stream Processing Algorithms remain active.

[b] Standard, alternate, and OEM Audio Stream Processing Algorithms are disabled. Custom Audio Stream Processing Algorithms remain active.

Note: ***Gear Limitation for PAY and PAZ***

Use of the Decode ASP Gear Control Register in PA is restricted to All, Standard, Cus, and Nil modes. Other modes are not currently available.

This restriction is a result of the fact that the Audio Stream Split Algorithm should not be disabled using the Decode ASP Gear Control Register in PA, as this PA Framework is not (yet) sophisticated enough to handle the resulting signal routing.

5.5.3 De-emphasis

The De-emphasis (DEM) Algorithm provides digital de-emphasis for linear PCM input signals. Automatic activation of de-emphasis occurs whenever it is detected that the input signal has been pre-emphasized. However, the provision for manual activation and deactivation of de-emphasis is also available. Default operation of the DEM Algorithm automatically activates de-emphasis for, and only for, stereo digital PCM input that is flagged as pre-emphasized.[35]

Alpha code for use with De-emphasis is given in Appendix C (De-emphasis Algorithm (DEM)).

5.5.4 Dolby Pro Logic IIx

Dolby Pro Logic IIx is a specific form of Audio Stream Processing that falls under the general classification of surround processing.

PA will, by default, perform Dolby Pro Logic IIx decoding on Dolby Digital surround-encoded stereo source signals and for all PCM bit streams. However, it can be turned off, or it can be turned on for other stereo source signals. The latter includes Dolby Digital two-channel bit streams that are either not surround encoded or for which no indication of surround encoding is indicated, as well as other types of 2-channel, encoded bit streams.

Details and alpha code for the Dolby Pro Logic IIx is given in corresponding IP package.

5.5.5 Dolby Digital EX

*Dolby Digital EX processing is not available with Feature Set*Dolby Digital EX is a specific form of Audio Stream Processing that falls under the general classification of surround processing.

PA will, by default, perform Dolby Digital EX decoding on Dolby Digital EX encoded signals and for signals for which there is no indication as whether or not they are Dolby EX encoded or not. However, it can be turned off, or it can be turned on for other stereo surround signals. The latter includes Dolby Digital bit streams that are not Dolby Digital EX encoded as well as other types of stereo surround-encoded bit streams.

Note: ***Dolby Digital EX***

The implementation provided with PA is Dolby Digital EX using Dolby Pro Logic IIx.

Alpha code for use with Dolby Digital EX is given in Dolby Pro LogicIIx IP package.

5.5.6 DTS Neo:6 Two-Channel Matrix

DTS Neo:6 Two-Channel Matrix is a specific form of Audio Stream Processing that falls under the general classification of surround processing.

PA will, by default, perform DTS Neo:6 Two-Channel Matrix processing on DTS encoded stereo source signals. If Pro Logic IIx is turned off, it will operate on PCM inputs. However, it can be turned off, or it can be turned on for other stereo source signals.

Alpha code for use with DTS Neo:6 Two-Channel Matrix is given in corresponding IP package.

[35] This statement is true for any *A003 (IROM3) based build. *A004 (IROM4) based builds will currently apply de-emphasis even for bitstream input when the pre-emphasis flag is indicated by the channel status data of the audio input signal.

5.5.7 DTS ES Matrix 6.1

DTS ES Matrix 6.1 is a specific form of Audio Stream Processing that falls under the general classification of surround processing.

PA can be configured to, by default, apply DTS ES Matrix 6.1 processing to DTS program material as part of the DTS Neo:6 Audio Stream Processing Algorithm. Additionally, it is possible to enable DTS ES Matrix 6.1 processing for non-DTS program material as part of Audio Stream Processing.

Alpha code for use with realization of DTS ES Matrix 6.1 via Audio Stream Processing is given in DTS Neo:6 IP package.

5.5.8 Matrix

The Matrix ASP (MTX) Algorithm is a surround processing algorithm that operates on a stereo input and expands the input to a soundfield of multiple output channels. The MTX algorithm includes the capability to fill satellite channels not available in the input to its output.

Alpha code for use with Matrix processing is given in Appendix E (Matrix).

5.5.9 Room Simulator

The Room Simulator Number 1 (RVB1) Algorithm is a surround processing algorithm that uses a proprietary stereo to quadraphonic reverberator to simulate playback of audio in rooms of various sizes and with different acoustic characteristics. The RVB1 algorithm includes the capability to fill satellite channels not available in the input to its output.

Alpha code for use with Room Simulator Number 1 processing is given in Appendix K (Room Simulator Number 1).

5.5.10 Downmix

The Downmix Algorithm Number 2 (DM) Algorithm provides a powerful form of downmix suitable for use with PA. In particular, it provides mixing capability that mimics that of the PCM Algorithm in many respects. Its primary use is as an adjunct to custom surround processing ASP Algorithms that do not themselves provide full downmix functionality as required.

Alpha code for use with Downmix processing is given in Appendix Q (Downmix Algorithm Number 2).

5.5.11 Graphic Equalization

The Graphic Equalizer (GEQ) Algorithm provides up to a 12-band graphic equalizer suitable for use with PA. In particular, the GEQ Algorithm can be used as a 2 to 12-band equalizer. Additionally, the GEQ Algorithm supports a 1-band, fixed frequency response Loudness Compensation (LOU) implementation.

The GEQ Algorithm handles changes to the level setting of individual bands in a manner that avoids audio artifacts.

Alpha code for use with Graphic Equalization processing is given in Appendix N (Graphic Equalizer).

5.5.12 Loudness Compensation

The Loudness Compensation ASP Algorithm provides the ability to apply a loudness compensating filter to the input signal on all channels.

Alpha code for use with Loudness Compensation processing is given in Appendix N (Graphic Equalizer).

5.5.13 Bass Management

PA provides a bass management system in order to:

- prevent speaker damage caused by sending full bandwidth audio to small speakers,
- take full advantage of the presence of a subwoofer, and
- ensure that all bass information in the input bit stream is reproduced properly regardless of speaker system configuration.

The bass management system in place on PA is based upon that recommended by Dolby, but comprehensive enough to encompass all potential speaker size combinations. For descriptions and diagrams of Dolby Output Configurations, please refer to the *Dolby Digital Licensee Information Manual, Issue 5*.

Complete details regarding the operation of Bass Management Number 2 is given in Appendix H (Bass Management Algorithm (BM2)).

The default mode of PA allows automatic selection of an appropriate output configuration based upon the speaker configuration control registers, and use of this mode is recommended in order to simplify operational control. This mode can be enabled and disabled both individually and in concert with other speaker configuration control. When automatic selection is enabled, the output configuration will be chosen automatically based upon the settings in the speaker configuration control registers and other system information, and the user may not select the bass management output configuration directly. When speaker configuration or automatic selection is disabled, the user must select the bass management output configuration directly.

Certification Requirement:

Dolby requires that the product have knowledge of the number and types of speakers attached to the product in order to deliver program material appropriately. A simple interface to control speaker configuration is discussed above in the speaker configuration registers. A more complex interface is also presented in Table 4-26 of section 4.11.1 "Speaker selection options" of the Dolby Digital Licensee Information Manual, Issue 5. *This interface is reproduced through the speaker configuration registers and the output configuration auto selection mode in PA. For this reason, it is recommended that the auto selection mode be used.*

CAUTION

Example 5-3 (Bass Management) below generates full-range signals at the subwoofer output that may damage speakers. Use care when working through this example.

EXAMPLE 5-3

Bass Management

This example demonstrates the operation of bass management.

This example uses full-range signals. See the caution statement given above.

Note: ***Test Signals from Dolby DVD Demo and Test Disc - Version 1.0***

It is highly recommended that Titles 70-72 on the Dolby DVD Demo and Test Disc - Version 1.0 be used for this example. If using the DVD interface on a TV screen, select 5.1 Channel Test Signals - Distortion. Select ***one of the following*** *as indicated in the text of this example:*

(i) LFE FS: *30 Hz full scale tone on only the LFE channel.*

(ii) 5x30Hz FS: *30 Hz full scale tone on only the satellite channels.*

(iii) 6x30Hz FS: *30 Hz full scale tone on LFE and satellite channels.*

If the Dolby DVD Demo and Test Disc is not available, use any other test disc that provides the appropriate signals.

Send Alpha code to select Digital input and Analog output.

Play from a Dolby Digital 5.1 source. Select the 5x30Hz FS signal as mentioned in the note above.

If the default speaker configuration is used, Dolby Output Configuration 1 will be automatically selected. This fact can be verified using alpha code `readBMOCSelectOCAuto`, which will generate the response `writeBMOCSelectOCAutoDOC1CorrelOff`. An audible signal should be present on the subwoofer output if the source material has bass content on any of the 5.1 input channels, and this bass content will be filtered out of the satellite channels.

If automatic output configuration selection is enabled, the alpha code sequence `writeBMOCSelectOCAutoNone, readBMOCSelectOCAuto` will generate the response `writeBMOCStatusOCAutoDOC1CorrelOff`. The system will ignore the request for no output configuration because Dolby Output Configuration 1 has been automatically selected based upon the speaker settings.

Disable automatic selection of bass management output configuration using alpha code writeSYSRecreationModeDirect.[36] Select Dolby Output Configuration 0 using alpha code writeBMOCSelectOCAutoDOC0CorrelOff. An audible signal should be present on the subwoofer output if the source material includes content on any of the 5.1 input channels, but in this case the content will not be filtered out of the satellite channels.

Select no output configuration using alpha code writeBMOCSelectOCAutoNone. An audible signal may be present on the subwoofer output if the source material includes content on the LFE channel, but in this case the bass content of the satellite channels will not be routed to the subwoofer but instead will be routed only to the output for that channel.

At this point, if desired, one may select the LFE FS signal from the test disc and observe the presence of the low frequency information at the subwoofer only. Or alternatively, one may select the 6x30Hz FS signal from the test disc and observe the presence of the low frequency information at the subwoofer and the satellite channels. If either of these selections is made, be sure to go back to the 5x30 Hz FS signal before proceeding further.

Stop the Dolby Digital 5.1 source. Select the default Dolby Output Configuration 1 using alpha code writeBMOCSelectOCAutoDOC1CorrelOff. View the status of automatic selection with the alpha code readBMOCStatusOCAuto. The response wroteBMOCStatusOCAutoNoneCorrelOffPLOff indicates that the selection for Dolby Output Configuration 1 has not been engaged, and the audio output will not have changed.

Restart the system to engage this selection by depressing play on the Dolby Digital 5.1 source. Again view the status of automatic selection with the alpha code readBMOCStatusOCAuto. The response wroteBMOCStatusOCAutoDOC1CorrelOff indicates that the selection for Dolby Output Configuration 1 has now been engaged. Now, as originally, an audible signal should be present on the subwoofer output if the source material includes bass content on any of the 5.1 input channels, and this bass content will not be duplicated on the satellite channels.

Play a PCM source, as from a CD. Dolby Pro Logic IIx will be active by default. View the status of the output configuration select using alpha code readBMOCStatusOCAuto. The response wroteBMOCStatusOCAutoDPC1CorrelOff indicates that the Pro Logic IIx version of Dolby Output Configuration 1 is now engaged, rather than the standard version.

5.5.14 Speaker Location Delay

The implementation of Speaker Location Delay provided with Feature Set of PA gives the minimum functionality as set forth in THX requirements at 96 kHz.[37] What this means is that, when operating at a sample rate of 96 kHz or less, delays can be implemented of up to 20 ms on the surround and back channels, and 10 ms on the front and subwoofer channels. No negative delay is provided.

[36] This alpha code sequence has other effects not described here.

[37] THX requirements for speaker location delay are not met when operating at 192 kHz.

The implementation of Speaker Location Delay provided with the other Feature Sets of PA give the minimum functionality as set forth in Dolby requirements at 96 kHz.[38] What this means is that, when operating at a sample rate of 96 kHz or less, delays can be implemented of up to 15 ms on the surround channels, 15 ms on the back channels, and 5 ms on the center channel. No delay can be applied to the left, right, and subwoofer channels. No negative delay on the center channel is provided.

With Feature Sets , the Speaker Location Delay is incorporated within the PCM Encode (PCE) algorithm.

Details regarding the operation of Speaker Location Delay are given in Appendix I (Speaker Location Delay).

Speaker Location Delay can be either *time-based* or *distance-based*. Example 5-4 (Delay Control) and Example 5-5 (Speaker Location Control) demonstrate time-based and location-based Speaker Location Delay respectively.

EXAMPLE 5-4

Delay Control

This example demonstrates time-based delay control.

Send Alpha code to select Digital input and Analog output.

Play from a Dolby Digital 5.1 source that has a 30 Hz sine wave on every channel.

Disable Bass Management using alpha code sequence `writeSYSRecreationModeDirect, writeBMOCSelectOCAutoNone`. Also, disable Dolby Digital EX using alpha code `writePL2XDEXUseDisable`. Attach an oscilloscope to analog center and left surround output signals.

View the default value of the delay units using alpha code `readDELUnit`, which will yield the result `writeDELUnitTimeMilleseconds`[39] indicating that the Delay Control Registers are used to specify speaker delay in units of milliseconds. View the default value of the delay in milliseconds for the Center and Left Surround Channels using alpha code `readDELDelayCntr, readDELDelayLSur`, which will yield the result `writeDELDelayCntrN(0),writeDELDelayLSurN(5)` indicating that the default value of the delays are 0 ms and 5 ms, respectively. A phase difference between the two signals will be observed using the oscilloscope, where the left surround signal is delayed 5 ms relative to the center signal.

Change the delay on the Left Surround Channel from its default value of 5 ms to 0 ms using alpha code `writeDELDelayLSurN(0)`. No phase difference will be observed between the two signals, indicating that the center and left surround signals are not delayed relative to each other.

[38] Dolby requirements for speaker location delay are not met when operating at 192 kHz.

[39] The actual response will be `writeDELUnitMillisecondsQ0`.

EXAMPLE 5-5

Speaker Location Control

This example demonstrates location-based delay control.

Send Alpha code to select Digital input and Analog output.

Play from a Dolby Digital 5.1 source that has a 30 Hz sine wave on every channel. Disable Bass Management using alpha code sequence `writeSYSRecreationModeDirect, writeBMOCSelectOCAutoNone`. Attach an oscilloscope to analog center and left surround output signals.

Set the delay units to location in feet using alpha code `writeDELUnitLocation-Feet`. Set seven Delay Control Registers to indicate that the satellite speakers are located 10 feet from the listener using alpha code as follows:

```
writeDELDelayLeftN(10)
writeDELDelayRghtN(10)
writeDELDelayCntrN(10)
writeDELDelayLSurN(10)
writeDELDelayRSurN(10)
writeDELDelayLBakN(10)
writeDELDelayRBakN(10)
```

No phase difference between the two signals will be observed using the oscilloscope, indicating that the center and left surround signals are not delayed relative to each other.

Change the Delay Control Registers to indicate that the center speaker is located 8 feet from the listener and that the surround speakers are located 5 feet from the listener using alpha code as follows:

```
writeDELDelayLSurN(5)
writeDELDelayRSurN(5)
writeDELDelayCntrN(8)
```

A phase difference between the two signals will be observed using the oscilloscope, where the left surround signal is delayed approximately 2.5 ms relative to the center signal.

If the phase difference between the center and left signals and then the left surround and left signals is measured, it will be see that the center signal is delayed approximately 3 ms relative to the left signal and that the left surround signal is delayed approximately 5 ms relative to the left signal, as desired.

5.6 Sample Rate Conversion

5.6.1 Primary Audio Stream Sample Rate Conversion

The implementation of synchronous rate conversion provided with PAY allows synchronous rate conversion at simple integer ratios to be applied to the Primary Audio Stream. In the default operating condition, the output of the Primary Audio Stream will not be affected.

Synchronous rate conversion on the Primary has no effect on the Secondary Audio Stream.

Details regarding the operation of synchronous rate conversion with PAY are given in Appendix K (Synchronous Rate Conversion Number 4).

Synchronous rate conversion is completed at two places within the Primary Audio Stream Processing Chain of PAY. Downsampling occurs earlier in the chain than Upsampling.

Alpha code symbols for use with the synchronous rate conversion algorithm is given in Appendix S. Alpha code symbols used in the examples on Synchronous Rate Conversion in this section is reproduced in Table 5-13 (Synchronous Rate Conversion Alpha Code). When directing these alpha codes towards the Primary Audio Stream, if downsampling is desired, alpha codes should be directed towards SRC_A, and if upsampling is desired, towards SRC_B. For example:

- To view the contents of the Synchronous Rate Conversion Rate Stream Register associated with the *downsampling* SRC Algorithm, use alpha code `readSRC_A_RateStream`.
- To view the contents of the Synchronous Rate Conversion Rate Stream Register associated with the *upsampling* SRC Algorithm, use alpha code `readSRC_B_RateStream`.

Note: ***Intended and Actual Output Sample Rates for PAY***

Observe from Table 5-13 the difference between alpha code `readSRCRateStream` *and* `readSRCSampleRate`*. Whereas the former indicates the* ***intended*** *output sample rate, the latter indicates the* ***actual output*** *sample rate. The intended and actual output sample rates may be different if such conversion is not possible in the current audio processing state.*

TABLE 5-13 **Synchronous Rate Conversion Alpha Code**

Register	Alpha Code[a]	Description
Rate Request[b]	readSRCRateRequest	Return what the current rate request is.
	writeSRCRateRequestMax192[c]	Select automatic determination of output rate so that the maximum output rate is 192 kHz.
	writeSRCRateRequestMax48[d]	Select automatic determination of output rate so that the maximum output rate is 48 kHz.
	writeSRCRateRequestMin32[e]	Select automatic determination of output rate so that the minimum output rate is 32 kHz.
	writeSRCRateRequestMin64	Select automatic determination of output rate so that the minimum output rate is 64 kHz.
Rate Stream	readSRCRateStream	Return intended sample rate ratio.
	wroteSRCRateStreamFull	Intended output is full-rate.
	wroteSRCRateStreamHalf	Intended output is half-rate.
	wroteSRCRateStreamDouble	Intended output is double-rate.
Sample Rate	readSRCSampleRate	Indicate actual output sample rate.

[a] The notation “SRC” here is to be replaced by “SRC_A_” for access to the first instantiation in the ASP Chain and by “SRC_B_” for access to the second instantiation in the ASP Chain when multiple instantiations are present in a single audio stream.

[b] Selection of the *rate* is a deferred control operation as described in Section K.1.2 (Rate Ratio). In certain cases, it may be necessary to control the percentage of the audio frame buffer utilized by the Decode Algorithm as described in Section 3.2.7 (Audio Decode Buffer Ratio).

[c] This is the default for the Primary Downsampling SRC Algorithm Instantiation.

[d] This is the default for the Secondary Downsampling SRC Algorithm Instantiation.

[e] This is the default for the Primary Upsampling SRC Algorithm Instantiation.

Since synchronous rate conversion can only be performed at simple integer ratios, a sample rate cannot be directly specified or requested. Instead, integer ratios, along with maximum and minimum rates can be requested. An integer ratio up or down sample the source by a factor of two or four. A minimum or maximum rate will cause the source to be up or down sampled by a factor of two or four, if necessary in order to meet the requirements.

EXAMPLE 5-6 **Primary Audio Stream Synchronous Rate Conversion, Upsampling**

This example is only applicable to PAY and PAZ.

This example demonstrates the operation of synchronous rate conversion in the Primary Audio Stream to provide multi-channel, high-rate ADC output without altering any portion of the internal processing. Initially, verify the default input and output sample rates. Then, enable synchronous rate conversion so that the output sample rate is two times the input sample rate.

Send Alpha code to select Digital input and Analog output.

Play from a 44.1 kHz PCM source, such as an audio CD. The output of the Primary Audio Stream will be 3/4.1 output at 44.1 kHz.

With synchronous rate conversion enabled for minimum 32 kHz output, as is the case in the default system operating mode, the output of the Primary Audio Stream will be 3/4.1 output at 44.1 kHz. View the contents of the Synchronous Rate Conversion Rate Stream Status Register using alpha code `readSRC_B_RateStream`. The response `wroteSRC_B_RateStreamFull` indicates that the output rate is the same as the input rate.

View the contents of the Synchronous Rate Conversion Rate Request Register using alpha code `readSRC_B_RateRequest`. The response will be `writeSRC_B_RateRequestMax192`, which is actually the same as `writeSRC_B_RateRequestMin32`, the default for this SRC Algorithm in the Primary Audio Stream. This can be verified by checking the file `P:\alpha\src_a.h` and observing that the alpha codes for both are the same.

To allow for upmixing by a factor of two, tell the decoder to output at half of the possible frame length, by using alpha code `writeDECBufferRatio2.` Enable synchronous rate conversion for 44.1 kHz PCM input using alpha code `writeSRC_B_RateRequestMin64, writeDECCommandRestart.` By using alpha code `writeSRC_B_RateRequestMin64,` the output sample rate is indirectly requested by specifying that it be no lower than 64 kHz. Therefore, the output of the Primary Audio Stream will be 3/4.1 output at 88.2 kHz produced by up-sampling the audio data by a factor of two immediately before encoding and output. Alpha code `writeDECCommandRestart` resets decode processing and causes the resolution of the deferred control operation due to sample rate conversion.

View the contents of the Synchronous Rate Conversion Rate Stream Status Register using alpha code `readSRC_B_RateStream`. The response `wroteSRC_B_RateStreamDouble` indicates that the output rate is double the input rate.

EXAMPLE 5-7 Primary Audio Stream Synchronous Rate Conversion, Downsampling

This example is only applicable to PAY and PAZ.

This example demonstrates the operation of synchronous rate conversion in the Primary Audio Stream to downsample two-channel, high-rate input before subsequent Audio Stream Processing.[40] Initially, verify the default input and output sample rates. Then, enable sample rate conversion so that the internal processing and output sample rate is 48 kHz at a maximum.

Send Alpha code to select Digital input and Analog output.

[40] Note that unlike the implementation provided for upsampling which is multi-channel, the implementation provided for downsampling is limited to two-channel, which allows processing of stereo and mono signals only. A request to downsample multi-channel input is ignored. For more information, see Appendix K (Synchronous Rate Conversion Number 4).

Play from a 96 kHz PCM source. The output of the Primary Audio Stream will be 3/4.1 output at 96 kHz.[41]

With synchronous rate conversion enabled for maximum 192 kHz input, as is the case in the default system operating mode, the output of the Primary Audio Stream will be 96 kHz. View the contents of the Synchronous Rate Conversion Rate Stream Status Register using alpha code `readSRC_A_RateStream`. The response `wroteSRC_A_RateStreamFull` indicates that the output rate is the same as the input rate.

View the contents of the Synchronous Rate Conversion Rate Request Register using alpha code `readSRC_A_RateRequest`. The response will be `writeSRC_A_RateRequestMax192`, the default for this SRC Algorithm in the Primary Audio Stream.

Enable synchronous rate conversion for 96 kHz PCM input using alpha code `writeSRC_A_RateRequestMax48, writeDECCommandRestart.` By using alpha code `writeSRC_A_RateRequestMax48`, the input sample rate is indirectly requested by specifying that it be no higher than 48 kHz. Therefore, 96 kHz PCM input will be down-sampled by a factor of two to produce 48 kHz audio data to be processed by the remainder of the Primary Audio Stream. View the contents of the Synchronous Rate Conversion Rate Stream Status Register using alpha code `readSRC_A_RateStream`. The response `wroteSRC_A_RateStreamHalf` indicates that for this SRC Algorithm the output rate is half the input rate. Alpha code `writeDECCommandRestart` resets decode processing and causes the resolution of the deferred control operation due to synchronous rate conversion.

To demonstrate that this control register setting has no deleterious effect on various forms of bit-stream input, play from various sources and observe the output. Input from a Dolby Digital 5.1 source will produce 3/4.1 output at 48 kHz. Input from a DTS 5.1 source will produce 3/4.1 output at 48 kHz. Input from a DTS 96/24 source will produce 3/2.1 output at 96 or 88.2 kHz.

5.6.2 Secondary Audio Stream Synchronous Rate Conversion

The contents of this section are only applicable to PAY and PAZ.

The implementation of synchronous rate conversion provided with PAY allows integer-ratio, synchronous rate conversion at simple integer ratios to be applied to the Secondary Audio Stream. In the default operating condition, the output of the Secondary Audio Stream will always be 48 kHz or less. This is achieved by automatically applying 2:1 synchronous rate conversion if the output of the Primary Audio Stream would be 64, 88.2, or 96 kHz, and by automatically applying 4:1 synchronous rate conversion if the output of the Primary Audio Stream would be 192 kHz.

Synchronous rate conversion on the Secondary has no effect on the Primary Audio Stream.

[41] An alternative to the 96 kHz PCM source, which may be difficult to obtain, is to use a DTS 96/24 bit stream with System Recreation Mode Stereo. In this case, the output will be 2/0.0 rather than 3/2.1. Be sure to restore the default System Recreation Mode Auto before using any other bit stream sources as described in the last paragraph of this example.

Synchronous rate conversion is the only ASP Algorithm in the Secondary Audio Stream Processing Chain. To direct alpha code at this ASP Algorithm, use any of the techniques described in Galfa->Help->User's Guide.

Details regarding the operation of synchronous rate conversion with PAY are given in Appendix K (Synchronous Rate Conversion Number 4).

EXAMPLE 5-8 **Secondary Audio Stream Synchronous Rate Conversion**

This example is only applicable to PAY and PAZ.

This example demonstrates the operation of synchronous rate conversion for the Secondary Audio Stream.

Send Alpha code to select Digital input and Analog output.

Play from a DTS 96/24 source. The output of the Primary Audio Stream will be 3/2.1 output at 96 kHz.

With synchronous rate conversion enabled for maximum 48 kHz output, as is the case in the default system operating mode, the output of the Secondary Audio Stream will be 2/0.0 output at 48 kHz produced by downmix and downsampling of the DTS ES 96/24 3/2.1 output of the DTS Decode Algorithm. View the contents of the Synchronous Rate Conversion Rate Stream Status Register using alpha code `readSRCRateStream`.[42]

Disable synchronous rate conversion for 96 kHz Decode Algorithm output using alpha code `writeSRCRateRequestMax96.`[43] For Rate selection to take effect, either restart the DTS 96/24 track, or switch to the Primary Audio Stream and enter alpha code `writeDECCommandRestart`. The output of the Secondary Audio Stream will be 2/0.0 output at 96 kHz produced by downmix of the DTS ES 96/24 3/2.1 output of the DTS Decode Algorithm. View the contents of the Synchronous Rate Conversion Rate Stream Status Register for the SRC Algorithm in the Secondary Audio Stream using alpha code `readSRCRateStream`.

5.7 Volume Control

The application interface for *Volume Control* provides a functional view of the audio operations of PA.

Decode and audio stream processing algorithms typically require gain corrections to be applied in varying degrees to each of the output channels. PA keeps track of all required gain corrections for each algorithm in all operating modes as part of its implementation of volume control, and it can apply them automatically.

[42] Since this instantiation of the SRC Algorithm is part of the Secondary Audio Stream, techniques like those described in Galfa->Help->User's Guide

[43] A similar stricture applies in order to write the value of this control register using this alpha code symbol.

Special considerations regarding PAY are mentioned below.

PAY: PAY provides two audio outputs, the primary output and the secondary output.[44] The material presented in this section is applicable to both audio streams in an equivalent manner.

5.7.1 Volume Representation

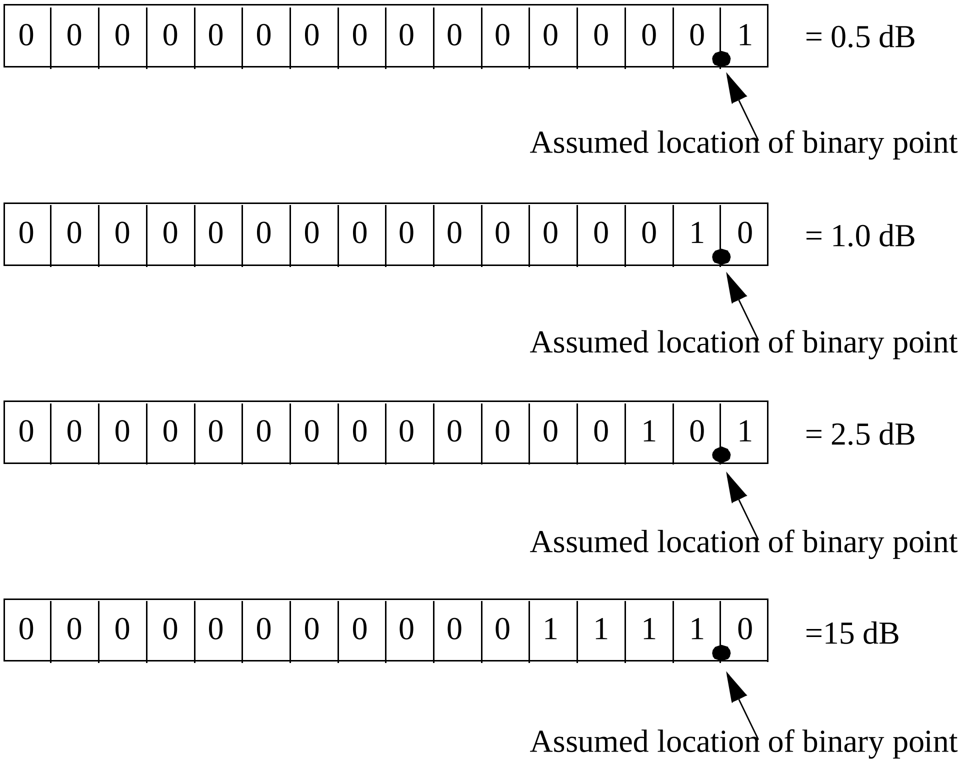
All volume control and status registers in PA use a common representation. All volume is represented using values in units of 0.5 dB. Values may be positive or negative. For example:

- To specify a value of +10 dB, the corresponding representation is actually +20 dB.
- To specify a value of -15 dB, the corresponding representation is actually -30 dB.

Typical volume control and status registers are 16-bit registers, but 8-bit registers are used as appropriate. Figure 5-10 (Volume Representation Using Q1 Values in dB) shows some examples using a 16-bit register with the Q1 notation.

[44] The primary output is generated by Audio Stream 1, and the secondary output is generated by Audio Stream 2.

FIGURE 5-10 **Volume Representation Using Q1 Values in dB**



5.7.2 Volume Control Implementation

Alpha code for volume control implementation is given in Table 5-14 (Alpha Code for Volume Implementation) and described below. Default settings are shown with a shaded background.

TABLE 5-14 **Alpha Code for Volume Implementation**

Register	Alpha Code	Description
Status	readVOLStatus	Return entire VOL status structure.
Mode	readVOLMode	Return Volume Mode Control Register value.
	writeVOLModeDisable	Disable volume operation.
	writeVOLModeEnable	Enable volume operation.

Register	Alpha Code	Description
Volume Control Implementation	readVOLImplementation	Return Volume Implementation Control Register value.
	writeVOLImplementationInactive	Volume control implementation is inactive.
	writeVOLImplementationInternal	Implement volume control internal to DSP.
	writeVOLImplementationExternal	Implement volume control external to DSP.

There are many different volume control implementations that can be used with PA, where the implementation used relates to the type of analog and digital level control hardware available in the system and the form of volume control desired by the operator. The volume control implementations listed in Table 5-14 (Alpha Code for Volume Implementation) are described below.

- The *Internal Volume Control Implementation* enables all facets of the volume control implementation. It is designed to be used when the volume control implementation is to be realized in the digital signal processing domain, completely internal to the DSP.
  - This volume control implementation is suitable for use with digital outputs or in low-end systems with analog outputs that lack hardware support for digital level control.
  - This is the default volume control implementation.
- The *External Volume Control Implementation* enables only some facets of the volume control implementation. It is designed to be used when the volume control implementation is to be realized external to the digital signal processing domain, completely or partially external to the DSP.

5.7.3 Volume Control Use

Alpha code for volume control use is given in Table 5-15 (Alpha Code for Volume Use) and is described in this section.

SeeTable 5-16 (Volume Control Channel Symbols) for a list of symbols appropriate for use to replace the meta-symbol *Channel* in Table 5-15. The location of the speakers corresponding to the volume control channel symbols in Table 5-16 are shown in Figure 5-1 (Speaker Location for Multi-channel Output).

TABLE 5-15 **Alpha Code for Volume Use**

Register	Alpha Code[a]	Description
Master Volume Control	readVOLControlMaster	Return current setting for the master volume.
	writeVOLControlMasterN(*N*)	Set master volume control. *N* is a signed, 16-bit value in units of 0.5 dB.
	writeVOLControlMasterN(-40)	Set master volume control to -20 dB.

Register	**Alpha Code**[a]	**Description**
Master Volume Offset	readVOLOffsetMaster	Return current limit set for the individual channel Volume Internal Status Registers.
	writeVOLOffsetMasterN(*N*)	Set upper limit for the individual channel Volume Status Registers. Despite the name, this is not an offset register.
	writeVOLOffsetMasterN(0)	Set the limit for the individual channel Volume Internal Status Registers to 0 dB.
Master Volume Internal	readVOLInternalStatusMaster	Indicates effective scaling to be applied as a result of the master volume control setting.
Master Volume External	readVOLExternalStatusMaster	Indicates 0. Reserved.
Volume Ramp Time	readVOLRampTime	Return Volume Ramp Time value in units of msec/dB.
	writeVOLRampTimeN(NN)	Set Volume Ramp Time in units of msec/dB. See writeVOLRampTimeN(50) below for an example. Volume Ramp Time determines the rate of gradual volume change from current Master Control Register setting to a new setting. The change is performed in linear steps, as indicated by units of msec/dB. This linear change is not indicated in the Master Volume Control Register. This is active for internal volume control implementations only.
	writeVOLRampTimeN(50)	Set Volume Ramp Time to 50 msec/dB. This equates to 20 dB/sec.
Channel Volume Control Trim	readVOLTrim*Channel*	Return offset from the Master Volume for the indicated channel.
	writeVOLTrim*Channel*N(*N*)	Set offset from the Master Volume for the indicated channel.
Channel Volume Internal	readVOLInternalStatus*Channel*	Indicate scaling to be applied internal to the DSP for the indicated channel.
Channel Volume External	readVOLExternalStatus*Channel*	Indicate scaling to be applied external to the DSP for the indicated channel.
Volume Control Implementation	writeVOLImplementationInternalDirect	Implement volume control internal to DSP with direct control of output level.
	writeVOLImplementationExternalDirect	Implement volume control external to DSP with direct control of output level.
	writeVOLImplementationN(*NN*)	Custom implementation of volume control.
Volume Control Channel Count	readVOLChannelCount	Indicate the number of channels under volume control.
	wroteVOLChannelCountN(8)[b]	Eight channels.

[a] SeeTable 5-16 (Volume Control Channel Symbols) for a list of symbols appropriate for use to replace the meta-symbol *Channel*.

[b] The default value is 8 for the primary audio stream but 2 for the secondary audio stream.

TABLE 5-16 **Volume Control Channel Symbols**

Symbol	Description
Left	Left Channel
Rght	Right Channel
LSur	Left Surround or Single Surround Channel
RSur	Right Surround Channel
Cntr	Center Channel
Subw	Single or Center Subwoofer Channel
LBak	Left Back or Single Back Channel
RBak	Right Back Channel
LSub	Left Subwoofer Channel
RSub	Right Subwoofer Channel

Before proceeding with the description of the volume control registers, it is important to keep in mind the following note:

Note: ***Volume Is "Set" Not Incremented***

One cannot change the volume setting in increments. One can only "set the volume" directly by providing an absolute volume.

Writing to a volume control register sets the volume level immediately
For internal volume control implementations, a volume ramp feature is provided. When enabled, this feature is applied each time the Master Volume Control Register is set. A gradual change from the current setting to the new setting is performed to produce a smooth volume change. This linear change is not indicated in the Master Volume Control Register.

The rate of the volume ramp, or time it takes to complete the smooth change, is adjustable by setting the Volume Ramp Time register.

Master Volume Control Register

The Master Volume Control Register sets the master volume level across all channels. In this register, an absolute volume is specified. The value 0 dB corresponds to the nominal level, that is, a full range input will produce a full range output. The default value -20 dB corresponds to a value 20 dB below this nominal value. See Example 5-9 (Volume Control) for a demonstration of basic volume control.

Certification Requirement:

> *Dolby requires a master volume control that will alter the volumes of all output channels uniformly while maintaining individual level relationships. Individual trim controls must be provided as well for each output channel in order to match speaker levels for proper sound balance. These trims help compensate for amplifier gains, speaker efficiencies, and room acoustics. See sections 4.10.4 "Master Volume Control" and 4.10.3 "Level trims and Balance Controls" of the* Dolby Digital Licensee Information Manual, Issue 5.
>
> *In addition, section 4.7.3 "LFE Level Trim Control" and section 4.7.4 "Subwoofer Level Trim Control" of the* Dolby Digital Licensee Information Manual, Issue 5*, suggests separate level controls for the LFE and the subwoofer. What distinguishes these two controls from each other is that the LFE level control alters the level of the program material on the LFE channel of the input prior to summing it with bass signals provided from the five main channels. The subwoofer volume level provides level control after the bass signals have been summed.*

Master Volume Offset Control Register

The Master Volume Offset Control Register is used not to provide any kind of offset in the volume level, but rather to prevent *misalignment* or *mistracking* by preventing any individual channel from getting "any louder" once any one of the individual channels has "reached" the maximum allowable volume level. In order that the master volume control maintain individual level relationships between the various channels, an upper limit on the value in any Volume Status Register is provided by the Master Volume Offset Control Register. The default value 0 dB prevents any channel volume from getting any higher once the Volume Status Register of any one channel has reached 0 dB. To effectively disable the prevention of mistracking, set the Master Volume Offset Control Register to its maximum value `0x7fff`. See Example 5-10 (Volume Mistracking) below.

Master Volume Internal and External Status Registers

The Master Volume Internal Status Register is normally the same value as the Master Volume Control Register. However, if the Master Volume Offset Control Register is set to a value that prohibits the master volume from achieving the level requested, the Master Volume Internal Status Register will instead be set to the "upper limit" of the Master Volume Control Register.

The Master Volume External Status Register has no current function.

Volume Trim Control Registers

The Volume Trim Control Registers for the individual channels specify the volume level for that channel relative to the master volume level. The default value 0 dB corresponds to the same level as the Master Volume, while a positive or negative value increases or decreases the relative volume level of that channel with respect to the Master Volume. See Example 5-9 (Volume Control) for a demonstration of mistracking, and for setting the Channel Volume Control Trim Register for the left channel.

The Subwoofer Volume Trim Control Register sets the volume level relative to the master volume level for the subwoofer channel at the output, that is, following bass management. To adjust the volume level for the LFE channel at the input, that is, preceding bass management, use the Bass Management LFE Volume Control Register as described in Appendix H (Bass Management Algorithm (BM2)).

Note: ***Certification Recommendation:***

Dolby recommends that volume trims have a limited range of -12 dB to +12 dB. To adhere to this recommendation, the Volume Trim Control Registers should be set only in the range -24 to +24 (as integral values).

Volume Internal Status Registers for Individual Channels

The Volume Status Registers for the individual channels are used as follows:

- The Volume Internal Status Registers for the individual channels reflect the gain applied in the digital signal processing domain, that is, within the DSP, to the audio data before output from the DSP. See Example 5-9 (Volume Control) and Example 5-10 (Volume Mistracking) below.
- The Volume External Status Registers for the individual channels reflect the gain to be applied in the analog signal processing domain, that is, without the DSP, to the audio data before output from the system.

In practical applications such as end-user systems, the noise generator is an ideal program source to use when setting individual volume trims. See Section 5.3 (Signal/Noise Generator) for a description of the Signal/Noise Generator.

Example 5-9 (Volume Control) and Example 5-10 (Volume Mistracking) demonstrate the volume control features described in this section.

Volume Control Implementation (Advanced Topic)

There are four facets to volume control implementation, enabled or disabled using the Volume Implementation Control Register as a bit-mapped register (as shown in Table 5-17 (Volume Implementation Control Register)):

- If the *Stream Facet* is enabled, then the desired output level is set according to the results of the volume computations derived as part of decoding and audio stream processing in conjunction with the settings present in the volume control registers.

  If the Stream Facet is disabled, then the desired output level is set directly using only the settings present in the volume control registers
- If the *Internal Facet* is enabled, then the desired output level derived according to the Stream Facet is set up to be realized using scaling internal to the DSP, that is, in the digital signal processing domain.

  If the Internal Facet is disabled, then the desired output level is set up to be realized using scaling external to the DSP, that is, in the analog signal processing domain.

- If the *Offset Facet* is enabled, then the scaling internal and external to the DSP is modified using the values in the Volume Control Offset Control Registers. That is, the scaling internal to the DSP is raised by the offset value for each channel, and the scaling external to the DSP is lowered by the offset value for each channel.

  If the Offset Facet is disabled, then the values in the Volume Control Offset Control Registers are not applied.
- If the *Apply Internal Facet* is enabled, then the scaling internal to the DSP is applied. If disabled, then the scaling internal to the DSP is not applied.

TABLE 5-17 **Volume Implementation Control Register**

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Reserved	Reserved	Reserved	Reserved	Apply Internal	Offset	Internal	Stream
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	*	*	*	*

These volume control implementation facets are used in combination to select a volume control implementation as follows:

- The *Inactive Volume Control Implementation* disables all facets of the volume control implementation by setting the Volume Implementation Control Register to 0x00. It is designed to be used when the volume control implementation is to be disabled.

  This volume control implementation is suitable for use in test and development but not in typical end-user systems.
- The *Internal Volume Control Implementation* enables all facets of the volume control implementation by setting the Volume Implementation Control Register to 0x0f. It is designed to be used when the volume control implementation is to be realized in the digital signal processing domain, completely internal to the DSP.

  This volume control implementation is suitable for use with digital outputs or in low-end systems with analog outputs that lack hardware support for digital level control.

  This is the default volume control implementation.
- The *Internal Direct Volume Control Implementation* enables appropriate facets of the volume control implementation to allow the internal volume control implementation to be “hijacked” and used to realize output levels under direct control of the operator. The Volume Implementation Control Register is set to 0x0e.

  This volume control implementation is suitable for use in test and development but not in typical end-user systems.
- The *External Volume Control Implementation* enables appropriate facets of the volume control implementation to allow the volume control implementation to be realized in the analog signal processing domain. The Volume Implementation Control Register is set to 0x0d.

  This volume control implementation is suitable for use in high-end systems with analog outputs that provide hardware support for digital level control. It is not suitable for use with digital outputs even in high-end systems.

- The *External Direct Volume Control Implementation* enables appropriate facets of the volume control implementation to allow the external volume control implementation to be “hijacked” and used to realize output levels under direct control of the operator. The Volume Implementation Control Register is set to 0x0c.

  This volume control implementation is suitable for use systems with analog audio paths that “bypass” the DSP but not the digital level control circuits.

Other combinations of these and other facets of volume control implementation can be used to provide custom volume control implementations.

Note: ***Implications of Volume Implementation***

The selection of a volume control implementation via enable or disable of the various facets has consequences in terms of the use of volume control.

5.7.3.1 Recommendation for usage of Volume Control algorithm.

This section will discuss the recommended usage of volume control algorithm by Audio equipment manufacturers for practical scenarios based on the capability of the *System*. Here, we mean the external volume control circuitry by the word *System*.

Let’s consider an example which will be discussed in the following paragraphs to understand the approaches. Let’s consider the input is 3/2/2 satellite channels and they are downmixed to 2/2 satellite channels. It means that downmixed output will not have Center and Back channels; they are downmixed to Left/Right (Front) and Left/Right (Surround) channel pairs, respectively. As a result of this downmix, it might be possible that output levels of Left and Right (Front) channels could go upto 4.5dB if Left, Center, Right channels were at 0.0dB at input. Similarly, Left/Right (Surround) channels can go upto 6.0dB. This would lead to saturation if enough care is not taken inside the volume algorithm or if the volume control algorithm is not configurred properly.

Now let’s see two different approaches to take care of this problem based on the capability of the *System*.

5.7.3.1.1 Internal Volume Control

This type of volume control is recommended to be used when the System is not capable of providing the gain/attenuation to individual channel and the System can provide only single gain/attenuation to all the output channels.

In this case, DSP needs to take care of individual level relationship between the output channels and also the output from DSP should not be saturated. Following are the alpha commands that are recommended to configure volume control algorithm.

```
writeVOLControlMasterN(0), writeVOLOffsetMasterN(0)
```

Micro can send `readVOLInternalStatusMaster` to DSP to know the common attenuation, for all channels, that need to be compensated by the ***System****. It’s suggested to ensure that volume has ramped completely before sending this alpha command.*

Explanation with example:

The saturation (in the downmix example) is automatically prevented by the setting, `writeVOLOffsetMasterN(0)`. However, with this setting, all channels are brought down by 6.0dB to maintain the balance of levels between the channels. Please note that Left/Right (Front) channels are also brought down by 6.0dB and not 4.5dB. Micro can decide to compensate this attenuation by setting up the *System* to provide this gain to all the channels.

5.7.3.1.2 External Volume Control

This type of volume control is recommended to be used when the system is capable of providing the gain/attenuation for individual output channel.

In this case, DSP is expected to provide the output with maximum signal-to-noise ratio without saturating the output. Following are the alpha commands that are recommended to configure volume control algorithm.

`writeVOLImplementationExternal`, `writeVOLControlMasterN(0)`, `writeVOLOffsetMasterN(0x7FFF)`

*Micro can send `readVOLExternalStatusChannel` to DSP to know the amount of attenuation, for every output channel, that need to be compensated by the **System**.*

Explanation with example:

Going by the downmix example, with the above volume control settings, Left/Right (Front) channels are automatically attenuated by 4.5dB and Left/Right (Surround) channels are automatically attenuated by 6.0dB, inside the DSP. Please note that output levels of Left/Right (Front) channels are not affected by the levels of Left/Right (Surround) channels, unlike when the volume control was set to internal. Micro can send the alpha commands for each output channel to know how much attenuation need to be compensated by the *System*. In this case, when Micro sends `readVOLExternalStatus-Channel` to all 2/2 output channels, it finds that Left/Right (Front) channels need 4.5dB and Left/Right (Surround) channels need 6.0dB gain to be applied by the *System*.

5.7.3.1.3 Volume Control with High Definition Audio decoders

High Definition audio decoders implement scaling of audio samples internally due to various reasons like speaker remapping, embedded downmixing etc. So, apart from following the approach discussed in 6.4.3.1.1 and 6.4.3.1.2, one needs to also do the following.

- Dolby Digital Plus:
  1. Enable speaker remapping by using `writeDDPSpeakerRemappingOn`
  2. Find scale factor by using `readDDPRemappingScaleFactor`.
  3. Apply the gain by the *System*.

Dolby Digital Plus alpha codes are described in its corresponding user’s guide. Description of the alpha code `readDDPRemappingScaleFactor` will help one understand how to calculate the gain value from this information.

- TrueHD:
  1. Speaker remapping is always enabled by default.
  2. Request decoder to apply scaling internally by the alpha command, `writeTHD-SamSizIgnore`.
  3. Find scale factor by using `readTHDRemappingScaleFactor`.

4. Apply the gain by the *System*.

TrueHD alpha codes are described in its corresponding user's guide. Description of the alpha code `readTHDRemappingScaleFactor` will help one understand how to calculate the gain values from this information.

In addition to the above, one also needs to take special care to compensate the gain, as indicated by `readTHDGainRequired, by the` *`System`*. More about this alpha command can be found in the corresponding user's guide.

- DTS-HD:
  1. Speaker remapping is always enabled by default.
  2. Find scale factor by using `readDTSHDBitStreamInformation26`.
  3. Apply the gain by the *System*.

DTS-HD decoder also has other operations where audio samples are scaled or affected.

Embedded Speaker Remapping: In this case, the speaker remapping information is already presented in the bit-stream and in such a case only the embedded coefficients are applied. So, outputs are never expected saturate and therefore there's no need to take care of it by the *System*.

Embedded Downmix: In this case, the downmix information is already presented in the bit-stream and in such a case only the embedded coefficients are applied. So, outputs are never expected saturate and therefore there's no need to take care of it by the *System*.

Downmix (not Embedded Downmix): The user can always request the decoder to output only the number of channels that he wishes to listen. This case would be automatically handled by any of the approaches discussed in 6.4.3.1.1 and 6.4.3.1.2.

However if it's chosen to select the SoftLimiting implementation of downmixing (by `writeDTSHDSoftLimiterModeEnable`), it's required to compensate the gain, as indicated by `readDTSHDSoftLimiterScale`, by the *System*.

5.7.3.1.4 Volume Control with Dolby Headphone2 (DH2) and Dolby Virtual Speaker2 (DVS2)

Output level of L/R channels is down by 6.0dB due to normal operation of DH2 and DVS2. Dolby recommends to provide this gain by the *System*. There are two cases of implementation as described below.

1. Output level of L/R channels is down by 6.0dB. But in this case, volume control algorithm has no information about this change in level. So, one has to make sure that 6.0dB gain is applied by the *System*.

2. DH2 and DVS2 provide an internal gain of 6.0dB to L/R channels (this is the default case). DH2 and DVS2 algorithms inform volume control algorithm about this internal gain. If any of the volume control approaches (discussed in 6.4.3.1.1 and 6.4.3.1.2) are followed, Micro will automatically come to know how much gain to be applied by the *System*.

5.7.3.1.5 Volume Control with Pro Logic IIx (PL2X) and NEO6

Output level of all channels, after surround processing, are down by 3.0dB. Dolby and DTS recommend to provide this gain by the *System* to all the output channels. There are two cases of implementation as described below.

1. After surround processing, all channels are down by 3.0dB. But in this case, volume control algorithm has no information about this change in level. So, one has to make sure that 3.0dB gain is applied by the *System*.

2. PL2X and NEO provide an internal gain of 3.0dB to all the surround decoded channels. PL2X and NEO inform volume control algorithm about this internal gain. If any of the volume control approaches (discussed in 6.4.3.1.1 and 6.4.3.1.2) are followed, Micro will automatically come to know how much gain to be applied by the *System*.

EXAMPLE 5-9

Volume Control

This example demonstrates basic volume control.

Send Alpha code to select Digital input and Analog output. Play a Dolby Digital 5.1 source, and observe the volume level on all channels simultaneously.

Set the master volume to -15 dB[45] using alpha code `writeVOLControlMasterN(-30)`. The output signal on all channels will increase. Return the master volume to its default value -20 dB using alpha code `writeVOLControlMasterN(-40)`. Decrease the volume 10 dB on the left channel alone using alpha code `writeVOLTrimLeftN(-20)`. The output on the left channel alone will drop. Finally, set the master volume to -30 dB using alpha code `writeVOLControlMasterN(-60)`. The output on all channels will drop, and the left channel will still be lower in level than any of the other channels.

Repeat the sequence of steps above. After each action, including both reset and the four write operations, review the contents of the master and left channel trim volume status registers using alpha code `readVOLInternalStatusMaster` and alpha code `readVOLInternalStatusLeft`, respectively.

As explained in Section 5.7.1 (Volume Representation), volume is represented using values in units of 0.5 dB. To understand the correspondence between the dB value in decimal, and the hexadecimal value returned by alpha code `readVOLInternalStatusMaster` or `readVOLInternalStatusLeft`, refer to Table 5-18 (Hexadecimal, Equivalent Representation, and Decimal Values in dB). The first column shows the hexadecimal value returned by alpha code `readVOLInternalStatusMaster` or `readVOLInternalStatusLeft`. Note that these are all negative numbers. The second column shows the equivalent representation corresponding to these negative numbers. The equivalent representation is obtained by directly converting the hexadecimal values in the first column to their equivalent decimal integers. The third column shows the decimal value of the equivalent representation. The decimal value is obtained by taking into account the fact that the equivalent representation is in units of 0.5 dB. Thus, the decimal value is half the value of the equivalent representation.

[45] As explained in Section 3.1.1 (Galfa Window), with the default system control register settings, the master volume will not go any higher than -15 dB.

TABLE 5-18

Hexadecimal, Equivalent Representation, and Decimal Values in dB

Hexadecimal dB	Equivalent Representation	Decimal dB
0xffe2	-30	-15
0xffd8	-40	-20
0xffc4	-60	-30
0xffb0	-80	-40

CAUTION

Example 5-10 (Volume Mistracking) below uses high-level signals that may damage speakers or your ears. Use extreme care when working through this example.

EXAMPLE 5-10

Volume Mistracking

This example demonstrates the volume mistracking prevention feature.

This example uses high-level signals. See the caution statement given above.

Send Alpha code to select Digital input and Analog output. Play a Dolby Digital 5.1 bit stream that has a full-scale sine wave on every channel, and observe the output level on all channels simultaneously.

Change the left channel volume trim from its default 0 dB to +10 dB using alpha code `writeVOLTrimLeftN(20)`. The output level on the left channel should increase. Set the master volume to -10 dB using alpha code `writeVOLControlMasterN(-20)`. The level on all the channels should increase, and the left channel level should still be higher than the other channels. Set the master volume to 0 dB using alpha code `writeVOLControlMasterN(0)`. The level on all the channels should not change. The full volume specified in the last alpha code sequence has not caused the output level on any channel to change, and mistracking has been prevented.

Use alpha code `writeVOLOffsetMasterN(0x7fff)` to disable the prevention of mistracking. This alpha code does this by setting the Master Volume Offset Control Register to its maximum value, which effectively disables the prevention of mistracking. The output on the left channel will be +10 dB above full range, and will appear distorted due to clipping. Mistracking along with distortion has occurred.

Repeat the sequence of steps above. After each action, including both reset and the four write operations, review the contents of the master[46] and left channel trim volume status registers using alpha code `readVOLInternalStatusMaster` and alpha code `readVOLInternalStatusLeft`, respectively.

To understand the correspondence between the dB value in decimal, and the hexadecimal value returned by alpha code `readVOLInternalStatusMaster` or `readVOLInternalStatusLeft`, refer to Table 5-19 (Return Values from readVOLInternalStatus-

Master and readVOLInternalStatusLeft). The first column lists the alpha code sent in each step of this example. The second column shows the return values after sending `readVOLInternalStatusMaster` and `readVOLInternalStatusLeft`. The third column indicates the decimal values that correspond to the return values in the second column.

TABLE 5-19 **Return Values from `readVOLInternalStatusMaster` and `readVOLInternalStatusLeft`**

Alpha Code	Return value from `readVOLInternalStatusMaster`, `readVOLInternalStatusLeft`	Decimal dB
`writeVOLTrimLeftN(20)`	0xffd8, 0xffec	-20, -10
`writeVOLControlMasterN(-20)`	0xffe2, 0xfff6	-15, -5
`writeVOLControlMasterN(0)`	0xffe2, 0xfff6	-15, -5
`writeVOLOffsetMasterN(0x7fff)`	0x0000, 0x0014	0, +10

5.8 MIPS Load

The MIPS Load ASP Algorithm allows processing load to be added to the system for test purposes. Increasing the MIPS load is accomplished by increasing the value in the Count Register of the MIPS Load ASP in the Audio Stream Processing Chain (the *Primary* Audio Stream Processing Chain for PAY). This value is a 16-bit unsigned integer and can be varied from `0x0000` to `0xffff`. Details regarding the operation of MIPS Load are given in Appendix J (MIPS Load Algorithm (ML0)).

The MIPS Load ASP Algorithm is generally located at the end of the Audio Stream Processing Chain:

Note: ***Location of the MIPS Load Algorithm***

For PAY and PAD, the MIPS Load ASP Algorithm is located at the end of the Primary Audio Stream Processing Chain.

Example 5-11 (Increasing or Decreasing the MIPS Load on the CPU) provides an introduction to increasing, and decreasing, the MIPS load.

[46] Reviewing the contents of the master volume status register after setting the master volume to -10 dB, will show that it does not go any higher than -15 dB. This is expected with the default system control register settings, and is explained in Section 3.1.1 (Galfa Window). However, after disabling the prevention of mistracking, reviewing the contents of this register will show that this limit is no longer applicable.

EXAMPLE 5-11

Increasing or Decreasing the MIPS Load on the CPU

This example provides a basic demonstration on how to increase, or decrease, the MIPS load (CPU load).

Depending on the desired topology, using Code Composer Studio, load and run `pa.out`.

Play a PCM source, as from a CD, and verify audio output. Select `CPU Load Graph` from the `DSP/BIOS` menu of Code Composer Studio to observe the MIPS load on the CPU.

Use alpha code `readMLCount` to verify that in the default configuration, the value in the ML Count Register is zero.

Change the value in the ML Count Register from `0x0000` to `0x00ff` by using alpha code `writeMLCountN(0x00ff)`. Observe the increase in the CPU load. Verify audio output.

Change the value in the ML Count Register to `0x05ff` by using alpha code `writeMLCountN(0x05ff)`. Observe further increase in the CPU load. Verify audio output.

Change the value in the ML Count Register back to its default value by using alpha code `writeMLCountN(0x0000)`.

For each topology, decreasing the CPU load is achieved by disabling, suspending, or rendering inactive some of the components of the system. Reduce the CPU load by using alpha code as follows:

PAI/PAH:

```
writeDECASPGearControlNil,writeDECCommandRestart

writeVOLImplementationInactive
```

PAY:

```
execPAYOutSecondaryNone

writeDECASPGearControlNil,writeDECCommandRestart

writeVOLImplementationInactive
```

Observe reduction in the CPU load. Verify audio output.

APPENDIX A Signal/Noise Generator Algorithm

The Signal/Noise Generator (SNG) Algorithm is a decode algorithm with an Application Interface (API) that permits one to exercise control over various aspects of its operation. This appendix explains the details of this component of the Performance Audio Framework (PA/F) API.

A.1 Songs and Channel Sets

The SNG Algorithm "decodes" *Songs* which are defined as modes of the algorithm as per the following:

- None - no output.
- NoiseWhite - white noise
- NoisePink - pink noise
- NoiseDolby - pink noise as defined by Dolby (satellites only)
- NoiseTI - pink noise (subwoofers only)
- NoiseTHX - noise with spectrum as defined by THX
- Tone - single-frequency sinusoid in one channel at a time
- Multitone - sinusoids in multiple channels simultaneously.

For additional information on *Songs*, see Chapter A.3 (Songs).

The SNG Algorithm operates in one of the following *Operational Modes*:

- Channel - produce output on a particular channel indefinitely.
- Sequence - sequence through individual channels, producing sound on one channel at a time.
- Cross-Channel - produce output on multiple channels simultaneously, indefinitely.

For additional information on *Operational Modes*, see *Chapter A.4 (Operational Modes)*.

The *satellite* and *subwoofer* channels make up two distinct *channel sets*. Some songs are applicable to these two channel sets separately, where it is appropriate that different types of signals be generated for each set. Some songs are applied across these two channel sets simultaneously. In general, the channel and sequence operational modes are applied separately, and the cross-channel operational mode is applied simultaneously.

It should be noted that Channel Configuration affects the Channel Set. The Channel Set is a subset of the Channel Configuration. For example, if the System Channel Configuration Request is Surround4_1 and the SNG Algorithm is in sequencing mode, it will sequence through LEFT, CNTR, RGHT, RSUR, RBAK, LBAK, LSUR, LEFT, etc. If, however, the System Channel Configuration Request is Surround3_1, the sequence will be LEFT, CNTR, RGHT, RSUR, LBAK, LSUR, LEFT, etc.

Separate control registers are provided for manipulation of parameters associated with signal/noise generation for these two channel sets when separate control is appropriate. This includes frequency, multiplier, and phase control. This does not include command, song, interval, or volume control.

A.2 Control, Status, and Command Registers

Table A-20 (SNG Algorithm Alpha Code for IOS Switching) and Table A-21 (SNG Algorithm Alpha Code) gives a list of alpha code symbols for the SNG Algorithm. For an explanation of the various forms of register usage, see Section 7.1 (Register Architecture).

Please keep the following in mind when reading this table:

- Default settings for control registers are shaded. Default settings result at power-up or reset.
- Bit-mapped registers allow the specification of multiple values simultaneously using the logical-or of any combination of the command or response for that register.
- Alpha code of type `write` is used to set values in registers, alpha code of type `read` is used to get values from registers, and alpha code of type `exec` is used to cause more complicated operations to occur.
- Numeric values for the preprocessor symbols shown in this table and appendix are provided in C language form in the alpha code symbol file `P:\alpha\sng_a.h`.

All SNG control and status registers that contain channel configuration information use the representation shown in Table 4-3 (Alpha Code for Channel Configuration Registers). The *XXX* in these alpha codes should be replaced with *SNG*.

TABLE A-20 **SNG Algorithm Alpha Code for IOS Switching**

Topology	Alpha Code	Description
PAD/PAI/PAY/PAZ	execProductNameDInSing execProductNameIInSing execProductNameYInSing execProductNameZInSing	Turn on noise generator. (Note, the alpha code `writeIBSampleRateOverride48000Hz` is sent as part of this IOS shortcut).
PAD/PAI/PAY/PAZ	execProductNameDIn*X* execProductNameIIn*X* execProductNameYIn*X* execProductNameZIn*X*	Turn off noise generator (where *X* is a valid input device other than *Sing*).

TABLE A-21 **SNG Algorithm Alpha Code**

Register	Alpha Code	Description
Status	readSNGStatus	Return entire SNG status structure.
	readSNGControl	Return control registers of SNG status structure.
Mode	readSNGMode	Indicate if the Signal/Noise Generator is enabled or not.
	writeSNGModeDisable	Disable SNG Algorithm operation.
	writeSNGModeEnable	Enable SNG Algorithm operation.
Channel	readSNGChannel	Indicate which channel is selected for signal/noise generation, i.e., which channel is the “current channel”.
	writeSNGChannel(N)	Select channel for signal/noise generation, i.e., specify which channel is the “current channel”.
Command	readSNGCommand	Return SNG Command Register value.
	writeSNGCommandNone	No command pending.
	writeSNGCommandReset	Reset channel to first in series.
	writeSNGCommandAdvance	Advance channel to next in series.
	writeSNGCommandPause	Pause channel during sequencing.
	writeSNGCommandContinue	Continue sequencing after pause.
Song	readSNGSong	Indicate which song (signal or noise) is currently enabled.
	writeSNGSongNone	Generate silence.
	writeSNGSongNoise**White**SatChannel	Generate white noise for satellites in channel mode on Left Channel.
	writeSNGSongNoise**White**SatChannelContinue	Generate white noise for satellites in channel mode on current channel.
	WriteSNGSongNoise**White**SatSequence	Generate white noise for satellites in sequence mode starting with Left Channel.
	writeSNGSongNoise**White**SatSequenceContinue	Generate white noise for satellites in sequence mode on current channel.

Register	Alpha Code	Description
	writeSNGSongNoise**White**SubChannel	Generate white noise for subwoofers in channel mode on LSub Channel.
	writeSNGSongNoise**White**SubChannelContinue	Generate white noise for subwoofers in channel mode on current channel.
	writeSNGSongNoise**White**SubSequence	Generate white noise for subwoofers in sequence mode starting with LSub Channel.
	writeSNGSongNoise**White**SubSequenceContinue	Generate white noise for subwoofers in sequence mode on current channel.
	writeSNGSongNoise**Pink**SatChannel	Generate pink noise for satellites in channel mode on Left Channel.
	writeSNGSongNoise**Pink**SatChannelContinue	Generate pink noise for satellites in channel mode on current channel.
	writeSNGSongNoise**Pink**SatSequence	Generate pink noise for satellites in sequence mode starting with Left Channel.
	writeSNGSongNoise**Pink**SatSequenceContinue	Generate pink noise for satellites in sequence mode on current channel.
	writeSNGSongNoise**Pink**SubChannel	Generate pink noise for subwoofers in channel mode on LSub Channel.
	writeSNGSongNoise**Pink**SubChannelContinue	Generate pink noise for subwoofers in channel mode on current channel.
	writeSNGSongNoise**Pink**SubSequence	Generate pink noise for subwoofers in sequence mode starting with LSub Channel.
	writeSNGSongNoise**Pink**SubSequenceContinue	Generate pink noise for subwoofers in sequence mode on current channel.
	writeSNGSongNoise**Dolby**SatChannel	Generate pink noise as specified by Dolby for satellites in channel mode on Left Channel.
	writeSNGSongNoise**Dolby**SatChannelContinue	Generate pink noise as specified by Dolby for satellites in channel mode on current channel.
	WriteSNGSongNoise**Dolby**SatSequence	Generate pink noise as specified by Dolby for satellites in sequence mode starting with Left Channel.
	writeSNGSongNoise**Dolby**SatSequenceContinue	Generate pink noise as specified by Dolby for satellites in sequence mode on current channel.
	writeSNGSongNoise**TI**SubChannel	Generate pink noise for subwoofers in channel mode on LSub Channel.
	writeSNGSongNoise**TI**SubChannelContinue	Generate pink noise for subwoofers in channel mode on current channel.
	writeSNGSongNoise**TI**SubSequence	Generate pink noise for subwoofers in sequence mode starting with LSub Channel.
	writeSNGSongNoise**TI**SubSequenceContinue	Generate pink noise for subwoofers in sequence mode on current channel.
	writeSNGSongNoise**THX**SatChannel	Generate pink noise as specified by THX for satellites in channel mode on Left Channel. This alpha code is available only for configurations of PA that include THX.

Register	Alpha Code	Description
	writeSNGSongNoise**THX**SatChannelContinue	Generate pink noise as specified by THX for satellites in channel mode on current channel. This alpha code is available only for configurations of PA that include THX.
	writeSNGSongNoise**THX**SatSequence	Generate pink noise as specified by THX for satellites in sequence mode starting with Left Channel. This alpha code is available only for configurations of PA that include THX.
	writeSNGSongNoise**THX**SatSequenceContinue	Generate pink noise as specified by THX for satellites in sequence mode on current channel. This alpha code is available only for configurations of PA that include THX.
	writeSNGSongNoise**THX**SubChannel	Generate pink noise as specified by THX for subwoofers in channel mode on LSub Channel. This alpha code is available only for configurations of PA that include THX.
	writeSNGSongNoise**THX**SubChannelContinue	Generate pink noise as specified by THX for subwoofers in channel mode on current channel. This alpha code is available only for configurations of PA that include THX.
	writeSNGSongNoise**THX**SubSequence	Generate pink noise as specified by THX for subwoofers in sequence mode starting with LSub Channel. This alpha code is available only for configurations of PA that include THX.
	writeSNGSongNoise**THX**SubSequenceContinue	Generate pink noise as specified by THX for subwoofers in sequence mode on current channel. This alpha code is available only for configurations of PA that include THX.
	writeSNGSongToneSatChannel	Generate tone for satellites in channel mode on Left Channel.
	writeSNGSongToneSatChannelContinue	Generate tone for satellites in channel mode on current channel.
	WriteSNGSongToneSatSequence	Generate tone for satellites in sequence mode starting with Left Channel.
	writeSNGSongToneSatSequenceContinue	Generate tone for satellites in sequence mode on current channel.
	writeSNGSongToneSubChannel	Generate tone for subwoofers in channel mode starting with LSub Channel.
	writeSNGSongToneSubChannelContinue	Generate tone for subwoofers in channel mode on current channel.
	writeSNGSongToneSubSequence	Generate tone for subwoofers in sequence mode starting with LSub Channel.
	writeSNGSongToneSubSequenceContinue	Generate tone for subwoofers in sequence mode on current channel.
	writeSNGSongMultiTone	Generate multiple tones in channel mode across all channels.

Register	Alpha Code	Description
Channel Configuration Program	readSNGChannelConfigurationProgram	Return SNG Channel Configuration Program Control Register value.
	writeSNGChannelConfigurationProgram*X*	Specify the channels included in the channel set, i.e., the channels for which SNG generates output.
	writeSNGChannelConfigurationProgramUnknown	Use the Decode Channel Configuration Program as the channel set, i.e., the channels for which SNG generates output.
Volume	readSNGVolume	Indicate current volume setting for SNG.
	writeSNGVolume(N)	Set reproduction volume as an 8-bit signed number in units of 0.5 dB. The default volume is 0 dB as described in Section A.2.4 (Volume).
Interval	readSNGIntervalMS	Indicate current sequencing interval time.
	writeSNGIntervalMS(N)	Set sequence interval as a 16-bit signed integer in ms. The default interval is 2 seconds as described in Section A.4.2 (Sequence Mode).
Satellite Frequency	readSNGFreqSatHz	Indicate current frequency for satellite channel tone generation.
	writeSNGFreqSatHz(N)	Set frequency for satellite tone generation as a 16-bit signed integer in Hz. The default satellite frequency is 500 Hz as described in Section A.3.3 (Tone Generation).
Subwoofer Frequency	readSNGFreqSubHz	Indicate current frequency for subwoofer channel tone generation.
	writeSNGFreqSubHz(N)	Set frequency for subwoofer tone generation as a 16-bit signed integer in Hz. The default subwoofer frequency is 50 Hz as described in Section A.3.3 (Tone Generation).
Satellite Frequency Multiplier	readSNGMultSatQ8	Return SNG Satellite Frequency Multiplier Control Register value.
	writeSNGMultSatQ8(N)	Set frequency multiplier for satellite multi-tone generation as a 16-bit Q8 number. The default multiplier is 1.25 as described in Section A.3.4 (Multi-Tone Generation).
Subwoofer Frequency Multiplier	readSNGMultSubQ8	Return SNG Subwoofer Frequency Multiplier Control Register value.
	writeSNGMultSubQ8(N)	Set frequency multiplier for subwoofer multi-tone generation as a 16-bit Q8 number. The default multiplier is 1.25 as described in Section A.3.4 (Multi-Tone Generation).
Satellite Phase	readSNGPhaseSatMS	Indicate current phase for satellite channel tone generation
	writeSNGPhaseSatMS(N)	Set phase for satellite tone generation as a 16-bit signed integer in ms. The default phase is 0 ms as described in Section A.3.3 (Tone Generation).
Subwoofer Phase	readSNGPhaseSubMS	Indicate current phase for subwoofer tone generation.
	writeSNGPhaseSubMS(N)	Set phase for subwoofer tone generation as a 16-bit signed integer in ms. The default phase is 0 ms as described in Section A.3.3 (Tone Generation).

A.2.1 Mode

Use of the SNG Mode Control Register is reserved.

A.2.2 Channel

The SNG Channel Select Register is used to select or indicate the channel used by the generator algorithm when only one channel is active.

The default channel is the Left Channel.

The valid values of *N* are as indicated in the file paftyp_a.h and the typical values are shown in Table A-22 (Typical Valid SNG Channel Definitions).

TABLE A-22 **Typical Valid SNG Channel Definitions**

***N*-value**	**Channel Description**
0	LEFT
1	RGHT
2	CNTR
8	LSUR
9	RSUR
10	LBAK
11	RBAK

Note: ***Select vs. Control or Status***

The SNG Channel Register is a Select Register rather than a Control Register or Status Register.

If the song is generated in sequence mode, this register cannot be used by the host to specify a value, since this register is used internally by the system to select and indicate the channel on which the noise or tone is generated. In this case, this is not a control register but rather a status register.

If the song is generated in channel mode, this register must be used by the host to specify the channel on which SNG output is to appear. In this case, this is a control register rather than a status register.

If the song is generated in cross-channel mode, or no song is generated, this register is not used.

Note that sending

- writeSNGSongNoiseXXXSatChannel[47], or

- writeSNGSongNoiseXXXSatSequence[48], or
- writeSNGSongToneSatChannel, or
- writeSNGSongToneSatSequence

and then immediately sending “writeSNGChannel(N)” might not be effective in setting the current channel to N. This is so because when SNG processes an alpha code listed above to select noise or tone generation, it resets the current channel to the Left (i.e., “first”[49]) channel. Depending upon when this happens relative to the channel setting via “writeSNGChannel(N)”, the current channel may end up set to either Left or N.

In such a case, the recommendation is to send the alpha code to select noise or tone generation and then poll for the response to readSNGCommand indicating 0 before sending “writeSNGChannel(N)” to select the SNG Channel where the noise or tone is to be generated.

The same recommendation applies in case of noise or tone generation in subwoofer channels if there is more than one subwoofer channel in the system.

No such precaution is necessary if the alpha code to select noise or tone generation is of a type that doesn’t reset the current channel to Left. These type of alpha codes are same as the above but with a suffix of “Continue”, such as:

- writeSNGSongNoiseXXXSatChannelContinue, or
- writeSNGSongNoiseXXXSatSequenceContinue, or
- writeSNGSongToneSatChannelContinue, or
- writeSNGSongToneSatSequenceContinue, or
- writeSNGSongMultiTone

A.2.3 Channel Configuration Program

The SNG Channel Configuration Program Control Register can be used to override the DECChannelConfigurationRequest register (whether that register is being controlled automatically or not). This SNG register operates in a manner similar to that of the PCM Channel Configuration Program Control Register:

- If the SNG Channel Configuration Program Control Register is set to a value *other* than *unknown* or *none*, then that setting is the channel configuration used for tone or noise generation. It is also the channel configuration of the audio stream at the head of the Audio Stream Processing Chain.

  In this case, output of the SNG Algorithm is passed on for audio stream processing with a channel configuration request (i.e., DECChannelConfigurationRequest) that may differ from the channel configuration of the stream. This allows many ASP Algorithms to operate in an active state. For example, if the SNG Channel Configuration Program Control Register is set to *stereo* and the DEC Channel Configuration Request Select Register (assuming that `SYSRecreationModeDont` is used) is *surround*, and

[47] where XXX is “White”, “Dolby”, “THX”, or “Pink”

[48] ibid.

[49] In this note, it is assumed that the system channel configuration includes the Left speaker.

ASPs are not disabled via writeDECASPGearControlNil, and Dolby Pro Logic IIx operation is enabled, then the PLIIx Algorithm will run to convert the stereo program to surround output.

- If the SNG Channel Configuration Program Control Register is set to the value *none*, then *None* is the channel configuration used for tone or noise generation, meaning that no audio is generated. It is also the channel configuration of the audio stream at the head of the Audio Stream Processing Chain.
- If the SNG Channel Configuration Program Control Register is set to the value *unknown*, then the DECChannelConfigurationRequest is used by SNG for tone or noise generation. This channel configuration is also the channel configuration of the audio stream at the head of the Audio Stream Processing Chain.

  In this case, output of the SNG Algorithm is passed on for audio stream processing with a channel configuration request that is identical to the channel configuration of the stream. This causes some ASP Algorithms to remain inactive. For example, if the SNG Channel Configuration Program Control Register is set to *unknown*, the DEC Channel Configuration Request Select Register (assuming that `SYSRecreationModeDont` is used) is *surround*, Dolby Pro Logic IIx operation is enabled, and Bass Management operation is enabled, then the PLIIx Algorithm will remain inactive while the BM1 Algorithm will remain active.

The default channel configuration program for the SNG Algorithm is *unknown*.

Changes to the SNG Channel Configuration Program Control Register are *deferred* operations, and hence may require writeDECCommandRestart for effect.

A.2.4 Volume

The SNG Volume Control Register is used to control the volume at which all signal or noise output is to be generated relative to its nominal output level.

The default volume is 0 dB. This indicates nominal output as follows:

- Uniform noise is generated in the range [-1.,1.).
- Sinusoids are generated at full scale [-1.,1.).

The SNG Volume Control Register controls the volume of the signal or noise at the source. The signal noise volume is subsequently affected by the Master Volume and Trim controls in the manner that those controls affect all source signals.

When producing Dolby noise, a SNGVolume(0) setting results in noise output at a level similar to that of a 1kHz tone at -30dBFS.

A.3 Songs

The signal and noise generation modes provided by the SNG Algorithm are known as *songs*. Four songs are currently provided by the SNG Algorithm. This algorithm is designed to be easily extensible as regards the addition of additional songs to the repertoire.

The default song is *None*.

A.3.1 None Generation

Zero-valued output is generated. This is suitable for use with ASP Algorithms that generate signals. This capability could be used to generate system clocking in lieu of input.

A.3.2 Noise Generation

Noise generation consists of signal generation and conditioning of wideband signals that are used to excite the speakers in a room. The signal conditioning occurs partially in the decode algorithm itself and partially in the audio stream processing that is applied to the output of the decode algorithm. These two forms of signal conditioning are discussed separately below.

A.3.2.1 Noise Generation Decode Processing

White or pink noise is generated in various modes. White noise is generated using a pseudo-random number generator. Pink noise is derived from white noise using filters specified by various certification authorities.

- White noise is generated using a pseudo-random number generator.

  This implementation uses the system `rand()` function.

- Dolby provides specifications for satellite noise generation.
- THX provides specifications for satellite, subwoofer, and LFE noise generation. The difference between subwoofer and LFE noise generation is described in Section A.3.2.2 (Noise Generation Audio Stream Processing) below.

A.3.2.2 Noise Generation Audio Stream Processing

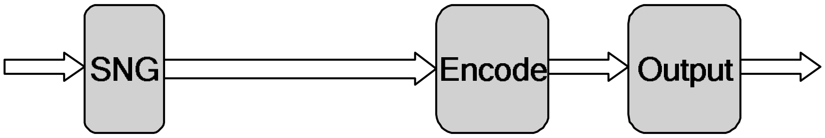
Two basic forms of noise generation are typical. One form of noise generation operates with Audio Stream Processing inactive, while a second operates with Audio Stream Processing active.

- Satellite noise generation typically occurs with Audio Stream Processing inactive.
- LFE noise generation typically occurs with Audio Stream Processing inactive.
- Subwoofer noise generation typically occurs with Audio Stream Processing active.

The operation of the SNG Algorithm for LFE and subwoofer noise generation is identical. What distinguishes the two is whether Bass Management is active or not. Either of the two types of bass noise generation can be selected using the SNG Channel Configuration Program Control Register as described in Section A.2.3 (Channel Configuration Program)However, the method for enabling LFE noise generation vs. subwoofer noise generation are different.

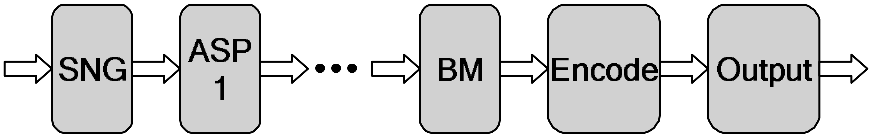
LFE noise generation is enabled by directly routing the LFE input to the subwoofer output, bypassing all Audio Stream Processing. This is easiest achieved by utilizing the Nil setting of the Decoder ASP Gear Control register (`writeDECASPGearControlNil`) followed by a resetting of the Decode state machine (`writeDECCommandRestart`) in order for the Gear Control modification to take effect. This is illustrated in Figure A-1 (SNG with Inactive Audio Stream Processing Chain (LFE Noise Generation)).

FIGURE A-1 **SNG with Inactive Audio Stream Processing Chain (LFE Noise Generation)**



Subwoofer noise generation is enabled by allowing all inputs to route through Bass Management. This is achieved by manually disabling all ASP Algorithms in the chain except Bass Management. This is illustrated in Figure A-2 (SNG with Active Audio Stream Processing Chain (SUBW Noise Generation)).

FIGURE A-2 **SNG with Active Audio Stream Processing Chain (SUBW Noise Generation)**



A.3.3 Tone Generation

A sinusoidal signal at the indicated frequency generated. The default frequency is 500 Hz for the satellite channels and 50 Hz for the subwoofer channels. The default phase is 0 ms (the phase of the signal is immaterial when using a single channel at a time).

The input sample rate to the SNG Algorithm (indicated by the IB Sample Rate Status register) is used to construct the tone for that sample rate. If the sample rate is not supported by the SNG Algorithm, there will be no tone output when presented with input at an unsupported sample rate. See Table C-5 (Sample Rate Support for SNG Algorithm) for information on supported sample rates for the SNG Algorithm.

A.3.4 Multi-Tone Generation

A sinusoidal signal at the frequency specified by the Satellite Frequency register, and at the phase specified by the Satellite Phase register, is generated in the first satellite channel (if any); the 'first' satellite channel is the lowest-numbered satellite channel in the channels identified by the DECChannelConfigurationRequest. Similarly, a sinusoidal signal at the frequency specified by the Subwoofer Frequency register, and at the phase specified by the Subwoofer Phase register, is generated in the first subwoofer channel (if any); the 'first' subwoofer channel is the lowest-numbered subwoofer channel in the channels identified by the DECChannelConfigurationRequest.

The second channel (both in the satellite and subwoofer channel sets) has a sinusoidal signal generated at frequency that differs from that of the first by the multiplier. That is, if the multiplier is 1.0, the frequency of the second sinusoid is equal to that of the first, if the multiplier is 0.5, the frequency of the second is equal to half that of the first, *etc*. The phase of the second channel sinusoid differs from that of the first by the phase. That is, the phase of the second channel sinusoid is twice that of the first channel sinusoid.

The third channel has a sinusoidal signal generated at frequency that differs from that of the second by the multiplier. That is, if the multiplier is 1.0, the frequency of the third sinusoid is equal to that of the second and to that of the first, if the multiplier is 0.5, the frequency of the third is equal to half that of the second and to one quarter that of the first, *etc*. The phase of the third channel sinusoid differs from that of the second by the phase. That is, the phase of the third channel sinusoid is thrice that of the first channel sinusoid.

The default multiplier is 1.25 for both the satellite and subwoofer channels.

> *The frequency of the second and subsequent sinusoids is rounded to integral Hz to facilitate implementation.*

The input sample rate to the SNG Algorithm (indicated by the IB Sample Rate Status register) is used to construct the tone for that sample rate. If the sample rate is not supported by the SNG Algorithm, there will be no tone output when presented with input at an unsupported sample rate. See Table C-5 (Sample Rate Support for SNG Algorithm) for information on supported sample rates for the SNG Algorithm.

A.4 Operational Modes

Three basic operational modes are provided that apply across songs. These basic operational modes are described below.

As the default song is None, the default operational mode is immaterial.

A.4.1 Channel Mode

In the *channel mode*, output is generated on a single channel for an indefinite period of time. The selection of the channel on which output is generated is under direct host control:

- The channel may be specified by the host directly using the SNG Channel Select Register.

  > *If the specified channel is not part of the current channel configuration, the output will be null.*

- The *reset* and *advance commands* may be used by the host to update the value of the SNG Channel Select Register.

  > *The reset and advance commands always produce a channel that is part of the current channel configuration. Thus, the output will never be null.*

- The reset command sets the channel to the “first” in the channel set. The first channel is the Left Channel for the Satellite Channel Set and the Left Subwoofer Channel for the Subwoofer Channel Set.[50]

- The advance command sets the channel to the “next” in the channel set. The “next” channel is considered clockwise around the room for the channels that are present in the requested channel configuration.

  If the specified channel from the SNG Channel Select Register is not part of the current channel configuration and the advance command is used, the output resumes with the next channel in the sequence. The default sequence is illustrated in Table A-23 (Default SNG Channel Sequence).

TABLE A-23 **Default SNG Channel Sequence**

N-value	From	*N*-value	To
0	LEFT	2	CNTR
1	RGHT	5	RWID
2	CNTR	15	RHED
3	not used	1	RGHT
4	LWID	14	LHED
5	RWID	7	ROVR
6	LOVR	4	LWID
7	ROVR	9	RSUR
8	LSUR	6	LOVR
9	RSUR	11	RBAK
10	LBAK	8	LSUR
11	RBAK	10	LBAK
12	LSUB	13	RSUB
13	RSUB	12	LSUB
14	LHED	0	LEFT
15	RHED	1	RGHT

The above table results in the following sequence order: LEFT, CNTR, RHED, RGHT, RWID, ROVR, RSUR, RBAK, LBAK, LSUR, LOVR, LWID, LHED, LEFT (for satellites) and LSUB, RSUB, LSUB (for subwoofers). Channels that are not requested are skipped.

- The *continue command* has no effect in *channel mode*.

[50] The first channel for the Satellite Channel Set is the Center Channel if the Left Channel is not part of the Satellite Channel Set, as is the case for monophonic output.

A.4.2 Sequence Mode

In the *sequence mode*, output is generated on a single channel for a specified time period. When the time period elapses, the channel on which output is generated automatically changes to the next channel. This continues from channel to channel in sequence around the room. The channel on which output is generated is not under direct host control:

- In this mode, the SNG Channel Select Register is used as a status register to indicate the channel that currently has noise being generated on it.
- The SNG Interval Control Register determines the time period for which output is generated on each channel. The default value is 2 seconds.
- The System Channel Configuration Request Select Register determines the sequence of channels on which output is generated. For `writeSYSRecreationModeAuto`, this occurs as part of the System Control Registers (e.g., the Speaker Control Registers or System Channel Configuration Request Registers). For `writeSYSRecreationModeDirect` or `writeSYSRecreationModeDont`, this occurs as part of the Channel Configuration Request Registers (e.g., the Decoder Channel Configuration Request Registers).
- The *reset*, *advance*, *pause*, and *continue commands* are used by the host to control the exact nature of the sequencing through channels.

  > *The result of these commands is always a channel that is part of the current channel configuration. The output will not be null unless there are no channels in that channel configuration.*

- The pause command stops the automatic sequencing indefinitely on the current channel. This produces the *paused state*.
- The continue command restarts the automatic sequencing from the current channel following a pause after a delay given by the interval. This command is not used except following a pause.
- The reset command sets the channel to the “first” in the channel set. The first channel is the Left Channel for the Satellite Channel Set and the Left Subwoofer Channel for the Subwoofer Channel Set.[51]
- The advance command sets the channel to the “next” in the channel set. The “next” channel is considered clockwise around the room for the channels that are present in the requested channel configuration.

  This command can be used in the sequence mode from a paused state to advance to the next channel and continue in the paused state. It may also be used while sequencing, in which case the channel will be reset to the next in the channel set immediately, where sequencing continues. Note that a delay between sending the advance command and checking to see what channel the noise is on may be needed.

A.4.3 Cross-Channel Mode

In this mode, output is generated across all channels simultaneously. There is no concept of a single channel or a sequence of channel states.

[51] *ibid*.

A.5 Initiation of Generation

Note: ***Using IOS Shortcuts***

In a typical system, the Signal/Noise Generator should be turned on/off by the user not via direct access to the Decode Source Select Control Register but rather indirectly via IOS Shortcuts of the form `execPAIInSing` *(for PAI). These shortcuts shall be designed to use the Decode Source Select Control Register, thus providing indirect control of the Signal/Noise Generator, in addition to clock control and other Signal/Noise Generation set-up control.*

The Signal/Noise Generator is turned on by setting the Decode Source Select Control Register to Sing after selecting None for the Decoder Source Select Register (`writeDECSourceSelectNone`. This is done with alpha code `writeDECSourceSelectSing`. In order to turn the Signal/Noise Generator off, this control register must be reset to some other source. This is best done via IOS Shortcuts.

As the default song is None, some song must be selected. Initial selection of a song typically takes one of two basic forms:

- Alpha code that simultaneously selects Noise Generation and resets the channel to the first in the series (*i.e.* `writeSNGSongNoiseSatChannel`).
- Alpha code that selects Noise Generation but does not reset the channel to the first in the series (*i.e.* `writeSNGSongNoiseSatChannelContinue`). Instead, Noise Generation is resumed either with the last active channel or a channel that has been set using the `writeSNGChannel(N)` alpha code.

These two basic forms apply in a like manner to Tone Generation. This distinction is not pertinent for Multi-Tone Generation, as this is a cross-channel operational mode.

The recommended alpha code sequence for enabling the Signal/Noise Generator (including one method of configuring the Signal/Noise Generator) is provided below:

- `writeDECASPGearControlNil` (to disable ASP chain)
- `writeDECCommandRestart` (to restart Decode state machine so that ASP chain disabling takes effect)
- execPA[DIYZ]InSing

  PAD: `execPADInSing`

  PAI: `execPAIInSing`

  PAY: `execPAYInSing`

  PAZ: `execPAZInSing`
- Begin sequenced satellite white noise generation using alpha code `writeSNGSongNoiseWhiteSatSequence`.
- Pause sequenced satellite white noise using alpha code `writeSNGCommandPause`.

- Continue sequenced satellite white noise generation using alpha code `writeSNG-CommandContinue`.

Stop Noise generation using the following alpha code sequence:

- `writeDECASPGearControlAll` (to enable ASP chain)
- `writeDECCommandRestart` (to restart Decode state machine so that ASP chain enabling takes effect)
- `execPA[IYD]In`*X* (where *X* is the desired audio input)

A.6 Sample Rates Supported by the SNG Algorithm

Table C-5 (Sample Rate Support for SNG Algorithm) lists the supported and unsupported sample rates for the SNG Algorithm.

TABLE A-24 **Sample Rate Support for SNG Algorithm**

Sample Rate	Supported?
32 kHz	Yes
44.1 kHz	Yes
48 kHz	Yes
88.2 kHz	Yes
96 kHz	Yes
192 kHz	Yes
64 kHz	Yes
128 kHz	Yes[a]
176 kHz	Yes[b]
8 kHz	No
11.025 kHz	No
12 kHz	No
16 kHz	No
22.05 kHz	No
24 kHz	No

[a] Only supported in

[b] *ibid.*

By default, SNG assumes that the output sample rate is 48kHz, so that SNG creates output samples consistent with a “playout” or “physical output sample” rate of 48kHz.

This default is effected by the definition of the execProductName[DIYZ]InSing shortcut, which includes invocation of ‘writeIBSampleRateOverride48000Hz’.

If a particular implementation were to use a playout rate other than 48kHz, it would be appropriate to use a writeIBSampleRateOverride which matches the actual playout rate.

The overall effect is:

- When outputting a tone, the observed tone frequency is as indicated in the following equation.
- When outputting noise or multitone, the spectrum is shifted similarly.

$$\text{tone frequency observed} = \text{tone frequency requested} * \left(\frac{\text{playout rate}}{\text{IBSampleRateOverride rate}}\right)$$

APPENDIX B PCM Decode Algorithm

The PCM Decode (PCM) Algorithm is a decode algorithm with an Application Interface (API) that permits one to exercise control over various aspects of its operation. This appendix explains the details of this component of the Performance Audio Framework (PA/F) API.

B.1 Control, Status, and Command Registers

Table B-1 (PCM Algorithm Alpha Code) gives a list of alpha code symbols for the PCM Algorithm. For an explanation of the various forms of register usage, see Section 4.1 (Register Architecture)

Please keep the following in mind when reading this table:

- Default settings for control registers are shaded. Default settings result at power-up or reset.
- Bit-mapped registers allow the specification of multiple values simultaneously using the logical-or of any combination of the command or response for that register.
- Alpha code of type `write` is used to set values in registers, alpha code of type `read` is used to get values from registers, and alpha code of type `exec` is used to cause more complicated operations to occur.
- Numeric values for the preprocessor symbols shown in this table and appendix are provided in C-language form in the alpha code symbol file `P:\alpha\pcm_a.h`.

The PCM Algorithm has no command registers.

TABLE B-1 PCM Algorithm Alpha Code

Register	Alpha Code	Description
Status	readPCMStatus	Return entire PCM status structure.
Control	readPCMControl	Return control registers of PCM status structure.
Mode	readPCMMode	Return PCM Mode control register.
	writePCMModeDisable	Disable PCM Algorithm.
	writePCMModeEnable	Enable PCM Algorihm.
Ramp	readPCMRamp	Indicate if PCM onset volume ramp is enabled or disabled.
	writePCMRampDisable	No PCM onset volume ramping - PCM starts at full volume.
	writePCMRampEnable	PVM onset volume ramping is enabled, PCM input will fade in when started.
Scale Volume	readPCMScaleVolume	Return the current volume scaling for PCM input.
	writePCMScaleVolumeN(0)	Set the volume scaling for PCM input to 0 dB.
	writePCMScaleVolumeN(*NN*)	Set the volume scaling for PCM input to *NN* dB in units of 0.5 dB.
LFE Volume	readPCMLFEDownmixVolume	Return PCM LFE Volume Control Register value.
	writePCMLFEDownmixVolumeN(2*10)	Set PCM LFE Volume Control Register to 10 dB.
	writePCMLFEDownmixVolumeN(*NN*)	Set PCM LFE Volume Control Register to *NN* dB in units of 0.5 dB.
Program Channel Configuration	readPCMChannelConfigurationProgram	Return PCM Program Channel Configuration Control Register value.
	writePCMChannelConfigurationProgramStereoUnknown	Interpret PCM input as unknown 2-channel material.
	writePCMChannelConfigurationProgram*X*	Interpret PCM input Channel Configuration as *X*. The *X* here is Channel Configuration as descibed in Table 4-3 (Alpha Code for Channel Configuration Registers)
Center Mix Level	readPCMCenterMixLevel	Return PCM Center Mix Level Register value.
	writePCMCenterMixLevelN(-2*3)	Set PCM Center Mix Level Register to -3 dB.
	writePCMCenterMixLevelN(*NN*)	Set PCM Center Mix Level Register to *NN* dB in units of 0.5 dB.
Surround Mix Level	readPCMSurroundMixLevel	Return PCM Surround Mix Level Register value.
	writePCMSurroundMixLevelN(-2*3)	Set PCM Surround Mix Level Register to -3 dB.
	writePCMSurroundMixLevelN(*NN*)	Set PCM Surround Mix Level Register to *NN* dB in units of 0.5 dB.
LFE Downmix Include	readPCMLFEDownmixInclude	Return PCM LFE Downmix Include Register value.
	writePCMLFEDownmixIncludeYes	Enable LFE Downmix operation when input contains LFE but LFE is not requested
	writePCMLFEDownmixIncludeNo	Disable LFE Downmix Operation irrespective of the request.

B.1.1 Ramp

By default, the PCM Algorithm applies an “onset ramp” to gradually increase the volume from zero to the nominal volume level described in Section B.1.2 (Scale Volume) when PCM input is initiated. This effect has been found to be aesthetically and aurally pleasing. The PCM Ramp Control Register can be used to disable or re-enable this capability.

The ramp-up occurs over 256 samples (1 block). At a sample rate of 44.1 kHz, that's 5.8 ms. The rate will vary slightly and appropriately for other sample rates.

There is no equivalent “offset ramp” currently implemented as part of the PCM Algorithm.

B.1.2 Scale Volume

The PCM Scale Volume Select Register is used to set a scale adjustment to be applied to all channels of PCM input. It is given in units of 0.5 dB (for example, the value 2*6 or 0x0c represents 6 dB). The default value of the PCM Volume Select Register is 0 dB for bit-stream input and 6 dB for analog input in PA.

This scale adjustment is designed for use in cases where the analog input is not at full scale due to hardware design limitations. Consider the case where the voltage swing of the input circuit is too small to accommodate a full scale input as in . The input is scaled down in hardware by a factor of 2 and scaled up internal to PA by 6 dB using the PCM Scale Volume Select Register to solve this design problem. This accounts for the default settings for PA.

This register is a select register rather than a control register. It is used by the System Stream to set alternate values for bit-stream and analog PCM input.

Note: ***PCM Scale Volume must be reset each time the input is switched.***

Switching between input modes (i.e. between digital and analog input) will reset the PCM Scale Volume to its default setting. If a different value is desired, this register must be reset after each input mode change.

There is no individual scale adjustment for the individual channels with PCM input.

B.1.3 LFE Downmix Volume

The PCM LFE Downmix Volume Control Register is used to set a gain to be applied to the LFE Channel of PCM input as part of downmix. It is given in units of 0.5 dB, so that the value 2*10 or 0x14 represents 10 dB. The default value of the PCM LFE Downmix Volume Control Register is 10 dB.A gain greater than 10 dB or less then 0 dB is not recommended.

One important caveat apply regarding application of the gain specified by the PCM LFE Downmix Volume Control Register:

- This gain is applied only as part of downmix that involves the LFE Channel. It is not applied to the LFE Channel at the PCM input in the general case.

B.1.4 Program Channel Configuration

The PCM Program Channel Configuration Control Register controls the interpretation of PCM input. Even though there are various possible settings for the PCM Program Channel Configuration Control Register (See Table 4-3 (Alpha Code for Channel Configuration Registers)), the following settings are most commonly used.

2-channel PCM input uses a Stereo settings for the PCM Program Channel Configuration Control Register:

1. Stereo Unknown: The input is either stereo or surround-encoded, but its exact nature is not known definitively.
2. Stereo Stereo: The input is stereo, that is, not surround-encoded.
3. Stereo Surround-encoded: The input is surround-encoded. That is, the input is either (1) Dolby Surround-Encoded and intended to be decoded using Dolby Pro Logic IIx decoding, or (2) DTS Surround-Encoded and intended to be decoded using DTS Neo:6 2-Channel Matrix.
4. Stereo Mono: The input is stereo mono, that is, a monophonic signal carried on two channels with a signal level 3 dB below that intended for reproduction using a single speaker.
5. Stereo Dual: The input is dual mono, that is, two monophonic signals carried on two channels.

5-channel PCM input uses a Surround2*_0 settings for the PCM Program Channel Configuration Control Register. The 5 channels are Left, Rght, Cntr, LSur, and RSur—no LFE. The format of the surround channels is specified as unknown, stereo, back-encoded, or monophonic.

6-channel PCM input uses a Surround2*_1 settings for the PCM Program Channel Configuration Control Register. The 6 channels are Left, Rght, Cntr, LSur, RSur, and LFE. The format of the surround channels is specified as unknown, stereo, back-encoded, or monophonic.

7-channel PCM input uses a Surround4*_0 settings for the PCM Program Channel Configuration Control Register. The 5 channels are Left, Rght, Cntr, LSur, RSur, LBak, and RBak—no LFE.

8-channel PCM input uses a Surround4*_1 settings for the PCM Program Channel Configuration Control Register. The 6 channels are Left, Rght, Cntr, LSur, RSur, LBak, RBak, and LFE.

*Note: Use of PCMChannelConfigurationProgramUnknown is **not** supported, even though the alpha command is defined. Different builds may produce different results if this alpha command is used.*

B.2 Downmix

Downmix of the PCM input as specified by the PCM Program Channel Configuration Control Register to the format requested by the PA Framework is provided. The PCM downmix follows the guidelines set out in *Digital Audio Compression (AC-3)*, Doc. A/52 Section 7.8. The following caveats are pertinent:

- For 2-channel PCM input, *Digital Audio Compression (AC-3)*, Doc. A/52 Sections 7.8.1 and 7.8.2 are equivalent except that the constant -3 dB is used (as per Section 7.8.1) rather than .7 (as per 7.8.2) in the case of a single surround channel.
- Mixing of the Back Channels is optional. See Section B.2.1 (Back Channel Downmix).
- Mixing of the LFE Channel is optional. See Section B.2.2 (LFE Channel Downmix).

B.2.1 Back Channel Downmix

For 5- and 6-channel PCM input, the Back Channels are not reproduced in the output or mixed into other channels as if they were additional Left and Right Surround channels. This follows A/52, which does not provide for input or mixing of the Back Channels.

For 7- and 8-channel PCM input, the Back Channels are reproduced in the output and are, when appropriate, mixed into other channels as if they were additional Left and Right Surround channels. This does not follow A/52, which does not provide for mixing of the Back Channels at a level other than 0.

B.2.2 LFE Channel Downmix

If the program does include an LFE Channel but the requested downmix does not include an LFE Channel, the LFE Channel can be mixed into Left and Right Channels.This mixing of LFE channels can be enabled or disabled using the PCM LFE Downmix Include Control Register.

When such operation is enabled, LFE is mixed into the Left and Right Channels at +7 dB to provide 10 dB boost with a two-channel reproduction correction (-3 dB).[52] This follows *Digital Audio Compression (AC-3)*, Doc. A/52 Section 7.8 which suggests such operation, but not Section 7.8.1 which does not include same.

Note that such mixing of the LFE Channel is optional in PA as suggested in *Digital Audio Compression (AC-3)*, Doc. A/52 Section 7.8. This mixing is controlled using the PCM Program Channel Configuration Control Register and the PCM LFE Downmix Include Control Register. If the LFE Downmix Include Control Register is set, the LFE Channel at the input is mixed into the output if it is included in the input (6- or 8-channel PCM input is selected), and the LFE Channel at the input is not mixed into the output if it is not included in the input (5- or 7-channel PCM input is selected).

Note: ***Downmix coefficients are un-normalized.***

The downmix coefficients used in the implementation of downmix are “un-normalized” in the sense described in Digital Audio Compression (AC-3), Doc. A/52 Section 7.8.1. In Performance Audio Frameworks such as PA where a floating-point representation of digital audio data is utilized, the

[52] In cases where the requested downmix does include the Center Channel but does not include the Left and Right Channels, the LFE Channel is instead mixed into the Center Channel at +10 dB to provide 10 dB boost without a multi-channel reproduction correction.

audio data is not scaled and so downmix coefficients are not scaled.

This scale information is not unimportant, however. Scale information is propagated throughout the PA Framework as the audio size, and it impacts the level and balance of the audio output from a system based upon PA.

B.2.3 Center and Surround Mix Levels

The parameters `clev` and `slev` are adjustable through the PCM Center Mix Level and PCM Surround Mix Level Register. These are 16 bit unsigned registers, dB value of which is in units of 0.5dB.

B.2.4 Decoder Bypass

The PCM algorithm supports Bypass functionality. This functionality allows for the data to be passed “as is” without any modification. The Bypass functionality can be enabled by using the alpha code writeDECBypassEnable. The Bypass functionality can be disabled by using the alpha code writeDECBypassDisable. By default, the Bypass functionality is disabled.

The Bypass functionality is considered useful in circumstances when it is required that the input fixed-point data should not be converted to floating point by the PCM algorithm. This is needed if some ASP algorithm requires fixed-point data input for processing. It is important that the ASP algorithm that requires fixed-point data input convert the data to floating point after the processing. This is to ensure that no ASP algorithm that requires floating-point data input by mistake receive fixed point data input.

If the Bypass functionality is enabled, some of the other functionalities of PCM algorithm are disabled. Specifically PCM ramp and PCM downmix functionality are disabled.

PCM Bypass functionality doesn’t affect the DECChannelMap features. Specifically if the DECChannelMapFrom for a particular channel is set to -1, the PCM algorithm will provide Zero-valued data for the channel. Similarly, if the DECChannelMapFrom for a particular channel is set to -2, no audio data is provided for the channel.

APPENDIX C De-emphasis Algorithm (DEM)

The De-emphasis (DEM) Algorithm is an Audio Stream Processing (ASP) Algorithm with an Application Interface (API) that permits one to exercise control over its operation. This appendix explains the details of this component of the Performance Audio Framework (PA/F) API.

The DEM Algorithm provides digital de-emphasis for linear PCM input signals. The control options provided allow automatic activation of de-emphasis whenever it is detected that the input signal has been pre-emphasized, as well as manual activation and deactivation of de-emphasis. Control options also allow selecting which channels de-emphasis may be active for. Default operation of DEM Algorithm automatically activates de-emphasis for pre-emphasized input signals and applies it to all active input channels.

Detection of whether or not an input signal has pre-emphasis is achieved by monitoring the pre-emphasis flags within the channel status data of the audio input signal (e.g., S/PDIF). This detection is performed by audio input processing, is not part of the DEM Algorithm, and is not described in this appendix. The result of pre-emphasis detection is conveyed as audio frame status to DEM Algorithm.

The DEM Algorithm may be available within multiple audio streams. The DEM Algorithm is present at the head of the ASP Chain associated with a particular audio stream.

C.1 Control, Status, and Command Registers

Table C-1 (De-emphasis Alpha Code) gives a list of alpha code symbols for the DEM Algorithm. For an explanation of the various forms of register usage, see Section 7.1 (Register Architecture).

Please keep the following in mind when reading this table:

- The commands defined for DEM Algorithm should not be interpreted as being required for selecting “normal” de-emphasis operation. Default operation automatically provides de-emphasis as appropriate for the input signal.

- Default settings for control registers are shaded. Default settings result at power-up or reset.
- Bit-mapped registers allow the specification of multiple values simultaneously using the logical-or of any combination of the command or response for that register.
- Alpha code of types `write` and `read` are used to set and get values to and from registers, respectively, and alpha code of type `exec` is used to cause more complicated operations to occur.
- Numeric values for the preprocessor symbols shown in this table and appendix are provided in C-language form in the alpha code symbol file `P:\alpha\dem_a.h`.
- As described above, there may be multiple instantiations of the DEM Algorithm for multiple audio streams. Register control for multiple instantiations of the DEM Algorithm in multiple audio streams must be directed at the appropriate audio stream using tools and techniques described elsewhere.

The DEM Algorithm has no command registers.

TABLE C-1 **De-emphasis Alpha Code**

Register	Alpha Code	Description
Status	readDEMStatus	Return entire DEM status structure.
Mode	readDEMMode	Return DEM Mode Control Register value.
	writeDEMModeDisable	Disable de-emphasis ASP.
	writeDEMModeEnable	Enable de-emphasis ASP.
Filter Mode	readDEMFilterMode	Return DEM Filter Mode Control Register value.
	writeDEMFilterModeOff	Select bypass of de-emphasis filter; de-emphasis filter is disabled. *This option is included for simple on/off control of the de-emphasis filter for MIPS measurements and any potential utility that it may serve in development and/or applications.*
	writeDEMFilterModeAuto	Select automatic activation of de-emphasis filter. De-emphasis filter is set to active/inactive based on the value of DEM Filter Active Status Register.
	writeDEMFilterModeOn	Select “forced-on” de-emphasis filter; de-emphasis filter is active. *This option is included to ease setup for testing of the de-emphasis filter.*
Filter Active	readDEMFilterActive	Return DEM Filter Active Status Register value. This register indicates whether de-emphasis filtering should be active or not, based on upstream processing. “Upstream” processing, such as input driver and decoder processing, determines whether or not de-emphasis should be applied. *Note that setting DEM Filter Mode Control Register has no effect on this register. This register continually indicates the proper on/off state for the De-emphasis filter, based on the current audio stream and upstream processing.*
	wroteDEMFilterActiveNo	Based on the current audio stream and upstream processing, de-emphasis filter should not be active.
	wroteDEMFilterActiveYes	Based on the current audio stream and upstream processing, de-emphasis filter should be active.

Register	Alpha Code	Description
Channel Enable	readDEMChannelEnable	Return DEM Channel Enable Control Register value. This is a bit-mapped register. Write alpha codes are provided below for logical "channel pairs" as well as custom channel selection.
	writeDEMChannelEnableNone	No channels are selected for de-emphasis.
	writeDEMChannelEnableLeftRght	Select only Left and Right channels for de-emphasis[a].
	writeDEMChannelEnableLSurRSur	Select only LSur and RSur channels for de-emphasis[b].
	writeDEMChannelEnableLBakRBak	Select only LBak and RBak channels for de-emphasis[c].
	writeDEMChannelEnableCntrSubw	Select only Cntr and Subw channels for de-emphasis[d].
	writeDEMChannelEnableAll	Select all channels for de-emphasis[e].
	writeDEMChannelEnableN(*NN*)	Set DEM Channel Enable Control Register to 0x*NN*, where 0x*NN* is a bit-mapped value. Setting a bit to '1' indicates the associated channel is selected for de-emphasis[f]. Bit map values for each channel are not provided here, however they can be extrapolated from the values of the writeDEMChannelEnable alpha codes above.
Channel Active	readDEMChannelActive	Return DEM Channel Active Status Register value. This is a bit-mapped register. This value indicates which channels are selected for de-emphasis, according to DEM Channel Enable Control Register setting and includes active channel(s) only. Indication of whether or not de-emphasis is applied to the selected channels is reported by `readDEMFilterActive`. Bit map values for each channel are not provided here, however they can be extrapolated from the values of the `wroteDEMChannelActive` alpha codes below. `wroteDEMChannelActive` alpha codes are provided for logical "channel pairs" only, similar to `writeDEMChannelEnable` alpha codes above, however the returned value is not limited to the values of these alpha codes as `writeDEMChannelEnableN(`*`NN`*`)` allows custom settings.
	wroteDEMChannelActiveNone	No channels are selected for de-emphasis.
	wroteDEMChannelActiveLeftRght	Left and Rght channels are active and selected for de-emphasis.
	wroteDEMChannelActiveLSurRSur	LSur and RSur channels are active and selected for de-emphasis.
	wroteDEMChannelActiveLBakRBak	LBak and RBak channels are active and selected for de-emphasis.
	wroteDEMChannelActiveCntrSubw	Cntr and Subw channels are active and selected for de-emphasis.
	wroteDEMChannelActiveAll	All channels are active and selected for de-emphasis.

[a] De-emphasis filter is only applied to active channels and according to DEM Filter Mode Control Register setting.

[b] ibid.

[c] ibid.

[d] ibid.

[e] ibid.

[f] ibid.

C.1.1 Mode

The DEM Mode Control Register controls basic operation of the DEM Algorithm. If zero, operation is disabled, and if non-zero, operation is enabled.

C.1.2 Filter Mode

The DEM Filter Mode Control Register controls how the de-emphasis filter is activated:

- While *Filter Mode Auto* is selected, the de-emphasis filter is automatically activated whenever it is detected that a PCM input signal has been pre-emphasized. This type of filter activation may be selected by sending the alpha code `writeDEMFilterModeAuto`. This is the default selection. Note that for bitstream input (e.g., Dolby Digital, DTS, etc.), with the *Filter Mode Auto* selection, the de-emphasis filter will not be activated based on the recommendation of Dolby and DTS.[53]
- While *Filter Mode Off* is selected, the de-emphasis filter is never activated. This type of filter activation may be selected by sending the alpha code `writeDEMFilterModeOff`.
- While *Filter Mode On* is selected, the de-emphasis filter is "forced on."This type of filter activation may be selected by sending the alpha code `writeDEMFilterModeOn`.

When active, the de-emphasis filter is applied to those channels reported by the DEM Channel Active Status Register.

Note: ***Filter Mode On and Filter Mode Off***

The purpose of providing the Filter Mode Off option is to allow simple on/off control of the de-emphasis filter for MIPS measurements and any potential utility that it may serve in development and/or applications.

The purpose of providing the Filter Mode On option is to ease setup for testing of the de-emphasis filter.

The DEM Filter Active Status Register reports whether de-emphasis filtering should be active or not, based on upstream processing. "Upstream" processing, such as input driver and decoder processing, determines whether or not de-emphasis should be applied. Setting DEM Filter Mode Control Register has no effect on this register. This register continually indicates the proper on/off state for the De-emphasis filter, based on the current audio stream and upstream processing:

- `wroteDEMFilterActiveYes`: Based on the current audio stream and upstream processing, de-emphasis filter should be active.

[53] This statement is true for any *A003 (IROM3) based build. *A004 (IROM4) based builds will currently apply de-emphasis even for bitstream input when the pre-emphasis flag is indicated by the channel status data of the audio input signal.

- `wroteDEMFilterActiveNo`: Based on the current audio stream and upstream processing, de-emphasis filter should not be active.

The de-emphasis filter response is as specified in Section C.1.3 (De-emphasis Filter Specification).

C.1.3 De-emphasis Filter Specification

The de-emphasis filter is a first-order, high-frequency-cut filter designed as the counterpart to the pre-emphasis processing commonly referred to as “50 microsec / 15 microsec“ or “CD type” pre-emphasis. The de-emphasis filter attenuates by about 2.4 dB at 3 kHz, increasing to about 7.6-dB attenuation at 10 kHz.

C.2 Operation

There are three modes of operation for de-emphasis in PA. These modes are described in this section and are associated with the three types of filter modes described in Section C.1.2 (Filter Mode).

C.2.1 Automatic De-emphasis (Default)

While *Filter Mode Auto* is selected, as described in Section C.1.2 (Filter Mode), DEM Algorithm provides automatic activation of de-emphasis whenever it is detected that a PCM input signal has pre-emphasis. This is the default operation. Note that for bitstream input (e.g., Dolby Digital, DTS, etc.), with the *Filter Mode Auto* selection, the de-emphasis filter will not be activated based on the recommendation of Dolby and DTS.[54]

During Automatic De-emphasis, the activation state of the de-emphasis filter is reported by the DEM Filter Active Status Register.

For an input signal that has been detected to have pre-emphasis, active de-emphasis filtering is reported as the alpha code `wroteDEMFilterActiveYes`. Detail of which active channel(s) have the de-emphasis filter applied is reported by the DEM Channel Active Status Register.

For an input signal that has not been detected to have pre-emphasis, de-emphasis filter is not active as reported by the alpha code `wroteDEMFilterActiveNo`.

The DEM Channel Enable Control Register allows selection of which channels may have de-emphasis filtering applied. The default setting of this register has all channels selected. For an input signal that has been detected to have pre-emphasis, the selected channels are evaluated for signal activity then the active channel(s) have the de-emphasis filter applied.

[54] This statement is true for any *A003 (IROM3) based build. *A004 (IROM4) based builds will currently apply de-emphasis even for bitstream input when the pre-emphasis flag is indicated by the channel status data of the audio input signal.

Note: ***DEM Channel Enable Control Register***

The DEM Channel Enable Control Register is provided for applications where it may be of interest to apply de-emphasis to a subset of channels for a given audio stream. For example, it may be of interest based on a priori *knowledge that it is desirable to apply (automatic or manual) de-emphasis to one linear PCM "channel pair" but not to a second pair.*

C.2.2 De-emphasis Off (Bypass)

While *Filter Mode Off* is selected, as described in Section C.1.2 (Filter Mode), DEM Algorithm remains active but the de-emphasis filter is bypassed. This effectively disables the DEM Algorithm, similar to using alpha code `writeDEMModeOff`, but without limitations of the latter. The limitation of using alpha code `writeDEMModeOff` is that DEM Algorithm operation is undefined if `writeDEMModeEnable` is issued afterward, and therefore this alpha code sequence should not be used.

While *Filter Mode Off* is selected, the DEM Algorithm can effectively be re-enabled by using alpha code `writeDEMFilterModeAuto` or `writeDEMFilterModeOn`.

The purpose of providing *De-emphasis Off* operation is to allow simple on/off control of the de-emphasis filter for MIPS measurements and any potential utility that it may serve in development and/or applications.

C.2.3 De-emphasis On (Test)

While *Filter Mode On* is selected, as described in Section C.1.2 (Filter Mode), DEM Algorithm applies de-emphasis filtering whether or not the input signal has pre-emphasis.

The DEM Channel Enable Control Register allows selection of which channels may have de-emphasis filtering applied. The default setting of this register has all channels selected. While *Filter Mode On* is selected, the selected channels are evaluated for signal activity then the active channel(s) have the de-emphasis filter applied.

The purpose of providing *De-emphasis On* operation is to ease setup for testing of the de-emphasis filter.

C.2.4 Usage of Audio Frame Data Structure Registers

A pointer to the Audio Frame Data Structure is passed to an ASP Algorithm as part of both the reset and apply function. It encapsulates a "frame" of audio sample data on which processing is to take place.

Table C-2 (Usage of Audio Frame Data Structure Quantities) illustrates the registers in the Audio Frame Data Structure which are utilized by the DEM Algorithm and whether the registers are read, written, or read *and* written.

TABLE C-2 **Usage of Audio Frame Data Structure Quantities**

Audio Frame Data Structure Register	Read/Written
channelConfigurationStream	R/W
sampleRate	R
sampleCount	R
data.sample	R/W
sampleProcess	W

The DEM field of the Sample Process register is set when DEM processing has successfully occured.

APPENDIX D Room Simulator Number 1

The Room Simulator Number 1 Algorithm is an Audio Stream Processing (ASP) Algorithm with an Application Interface (API) that permits one to exercise control over various aspects of its operation. This appendix explains the details of this component of the Performance Audio Framework (PA/F) API.

The Room Simulator Number 1 (RVB1) Algorithm uses a proprietary stereo to quadraphonic reverberator to simulate playback of audio in rooms of various sizes and with different acoustic characteristics. The RVB1 algorithm includes the capability to fill satellite channels not available in the input to its output.

D.1 Control, Status, and Command Registers

Table D-1 gives a list of alpha code symbols for the RVB1 Algorithm. For an explanation of the various forms of register usage, see Section 6.1 (Register Architecture).

Please keep the following in mind when reading this table:

- Default settings for control registers are shaded. Default settings result at power-up or reset.
- Alpha code of types `write` and `read` are used to set and get values to and from registers.
- Numeric values for the preprocessor symbols shown in this table and appendix are provided in C-language form in the alpha code symbol file `P:\alpha\rvb_a.h`.

The RVB1 Algorithm has no command registers.

TABLE D-1 **Room Simulator Number 1 Algorithm Alpha Codes**

Register	Alpha Code	Description
Status	readRVBStatus	Return entire RVB1 Algorithm status structure.
Mode	readRVBMode	Indicate if Room Simulator Number 1 Algorithm is enabled or disabled.
	writeRVBModeDisable	Disable the Room Simulator Number 1 Algorithm
	writeRVBModeEnable	Enable the Room Simulator Number 1 Algorithm
Bypass	readRVBBypass	Return the value of the Bypass Control Register.
	writeRVBBypassEnable	Disable the Room Simulator operation.
	writeRVBBypassDisable	Enable the Room Simulator operation.
Parameter Change	readRVBparamChg	Return the current value of the parameter change register.
	writeRVBparamChgDyn	Indicate that one or more of the dynamic parameters of the RVB1 Algorithm have been changed.
	writeRVBparamChgStat	Indicate that one or more of the static parameters of the RVB1 Algorithm have been changed.
	writeRVBparamChgPreset	Indicate that the Preset Register of the RVB1 Algorithm has been changed.
	writeRVBparamChgAll	Indicate that any of the parameters of the RVB1 Algorithm have been changed.
Effects Mix	readRVBfxMix	Return the value of the Effects Mix Register.
	writeRVBfxMix(N)	Set the value of the Effects Mix Register to N, a 16 bit value. Supported values are 0 to 100. 0 indicates dry mix and 100 a wet mix.
	writeRVBfxMix(50)	Set the Effects Mix Register as 50.
Reverberation Time	readRVBreverbTime	Return the value of the reverberation time (in hundreds of milliseconds).
	writeRVBreverbTime(N)	Set the reverberation time to N milliseconds. N is a 16 bit value. Valid values are 500-30000.
	writeRVBreverbTime(2500)	Set the reverberation time to 2.5 s.
Damping Frequency	readRVBdamping	Return the value of the Damping Frequency Register.
	writeRVBdamping(N)	Set the value of the Damping Frequency Register to N Hz. N is a 16 bit value.
	writeRVBdamping(4000)	Select Damping Frequency to be 4 kHz.
Front Input Gain	readRVBfrontInGain	Return the value of the Front Input Gain Register.
	writeRVBfrontInGain(N)	Set the input gain for the front channels (Left and Right) to N, a 16 bit value. N is interpreted as the gain amount scaled by 2^{14}.
	writeRVBfrontInGain(0x4000)	Set the Front Input Gain to 1.
Center Input Gain	readRVBcenterInGain	Return the value of the Center Input Gain Register.
	writeRVBcenterInGain(N)	Set the input gain for the center channel to N, a 16 bit value. N is interpreted as the gain amount scaled by 2^{14}.
	writeRVBcenterInGain(0x4000)	Set the Center Input Gain to 1.

Register	Alpha Code	Description
Surround Input Gain	readRVBsurrInGain	Return the value of the Surround Input Gain Register.
	writeRVBsurrInGain(N)	Set the input gain for the front channels (Left and Right Surround and Back channels) to N, a 16 bit value. N is interpreted as the gain amount scaled by 2^{14}.
	writeRVBsurrInGain(0x4000)	Set the Surround Input Gain to 1.
Front Output Gain	readRVBfrontOutGain	Return the value of the Front Output Gain Register.
	writeRVBfrontOutGain(N)	Set the output gain for the front channels (Left and Right) to N, a 16 bit value. N is interpreted as the gain amount scaled by 214.
	writeRVBfrontOutGain(0x4000)	Set the Front Output Gain to 1.
Center Output Gain	readRVBcenterOutGain	Return the value of the Center Output Gain Register.
	writeRVBcenterOutGain(N)	Set the output gain for the center channel to N, a 16 bit value. N is interpreted as the gain amount scaled by 214.
	writeRVBcenterOutGain(0x4000)	Set Center Output Gain to 1.
Surround Output Gain	readRVBsurrOutGain	Return the value of the Surround Output Gain Register.
	writeRVBsurrOutGain(N)	Set the output gain for the front channels (Left and Right Surround and Back channels) to N, a 16 bit value. N is interpreted as the gain amount scaled by 214.
	writeRVBsurrOutGain(N)	Set Surround Output Gain to 1.
Room Size	readRVBroomSize	Return the value of the Room Size Register.
	writeRVBroomSize(N)	Set the value of the Room Size Register to N, a 16 bit value. Valid values are 0-100, with 0 indicating the smallest simulated room and 100, the largest.
	writeRVBroomSize(50)	Set the value of the Room Size Register to 50.
Low Frequency Cut	readRVBLFCut	Return the value of the Low Frequency Cut Register
	writeRVBLFCut(N)	Set the value of the Low Frequency Cut Register to N, a 16 bit value. Valid values are 20-200, which indicate the cutoff frequency of the high-pass shelf filter in Hz.
	writeRVBLFCut(20)	Set the value of the Low Frequency Cut Register to 20 Hz.
Pre-Delay	readRVBPreDelay	Return the value of the Pre-Delay Register.
	writeRVBPreDelay(N)	Set the Pre-Delay to N/100 milliseconds. N is a 16 bit value. Valid values are 0-85.
	writeRVBPreDelay(0)	Set the value of the Pre-Delay Register to 0 ms.
Surround Delay	readRVBSDelay	Return the value of the Surround Delay Register.
	writeRVBSDelay(N)	Set the Surround Delay to N/100 milliseconds. N is a 16 bit value. Valid values are 0-85.
	writeRVBSDelay(0)	Set the value of the Surround Delay Register to 0 ms.

Register	Alpha Code	Description
Preset Room	readRVBloadPreset	Return the value of the Preset Room Register.
	writeRVBloadPreset(N)	Select the Nth preset room, where N is a 16 bit value. Valid values for N are 0-17.
	writeRVBBigHall	Select the “Big Hall” preset room. (Preset room number 0)
	writeRVBBrightHall	Select the “Bright Hall” preset room. (Preset room number 1)
	writeRVBDarkHall	Select the “Dark Hall” preset room. (Preset room number 2)
	writeRVBCathedral	Select the “Cathedral” preset room. (Preset room number 3)
	writeRVBClub	Select the “Club” preset room. (Preset room number 4)
	writeRVBBrightClub	Select the “Bright Club” preset room. (Preset room number 5)
	writeRVBSmokyClub	Select the “Smoky Club” preset room. (Preset room number 6)
	writeRVBBrightStadium	Select the “Stadium” preset room. (Preset room number 7)
	writeRVBArena	Select the “Arena” preset room. (Preset room number 8)
	writeRVBVoxCinema	Select the “Vox Cinema” preset room. (Preset room number 9)
	writeRVBMusicCinema	Select the “Cinema” preset room. (Preset room number 10)
	writeRVBBrightSpace	Select the “Bright Space” preset room. (Preset room number 11)
	writeRVBBiggerHall	Select the “Bigger Hall” preset room. (Preset room number 12)
	writeRVBBigArena	Select the “Big Arena” preset room. (Preset room number 13)
	writeRVBBiggestStadium	Select the “Biggest Stadium” preset room. (Preset room number 14)
	writeRVBJazzHall	Select the “Jazz Hall” preset room. (Preset room number 15)
	writeRVBSlapHappyClub	Select the “Slap Happy Club” preset room. (Preset room number 16)
	writeRVBSurroundClub	Select the “Surround Club” preset room. (Preset room number 17)

D.1.1 Mode

The RVB1 Mode Control Register controls the basic operation of the RVB1 Algorithm. If zero, operation is disabled, and if non-zero, operation is enabled if the RVB1 Bypass Control Register is disabled.

D.1.2 Bypass Control

The RVB1 Bypass Control Register provides the ability to enable and disable the RVB1 Algorithm.

If the Bypass Control Register is set to writeRVBBypassEnable then the RVB1 Algorithm *is not* applied to the input.

If the Bypass Control Register is set to writeRVBBypassDisable then the RVB1 Algorithm *is* applied to the input.

D.1.3 Parameter Change

The RVB1 Parameter Change Register is used to communicate changes in Status Registers to the RVB1 Algorithm.

When one of the *dynamic* parameters of the RVB1 Algorithm are changed, writeRVB-paramChgDyn must be used for the changes to take effect. Effects Mix, Reverberation Time, Room Size, Damping Frequency, and Low Frequency Cut Registers control the dynamic parameters of the RVB1 algorithm. Status registers of the RVB1 Algorithm associated with its dynamic parameters may be changed during audio playback without the risk of introducing audio artifacts.

When one of the *static* parameters of the RVB1 Algorithm are changed, writeRVBparam-ChgStat must be used for the changes to take effect. The Input Gains, Output Gains and Room Size Registers control the static parameters of the RVB1 Algorithm. Changing the value of the status registers of the RVB1 Algorithm associated with its static parameters during audio playback may introduce small clicks in the output audio. It is recommended that playback is muted before a static parameter change is flagged (via writeRVBparam-ChgStat) to the algorithm. Audio can be unmuted after the parameter change has been effected by the algorithm.

When the Preset Room Register of the RVB1 Algorithm is changed, writeRVBparamChg-Preset must be used for the new preset to take effect. Changing the preset room during audio playback may result in small clicks in the output audio. Precautions outlined in the previous paragraph are advised when changing the Preset Room Register.

writeRVBparamChgAll may be used to indicate that possibly both dynamic and static parameters of the RVB1 Algorithm have changed.

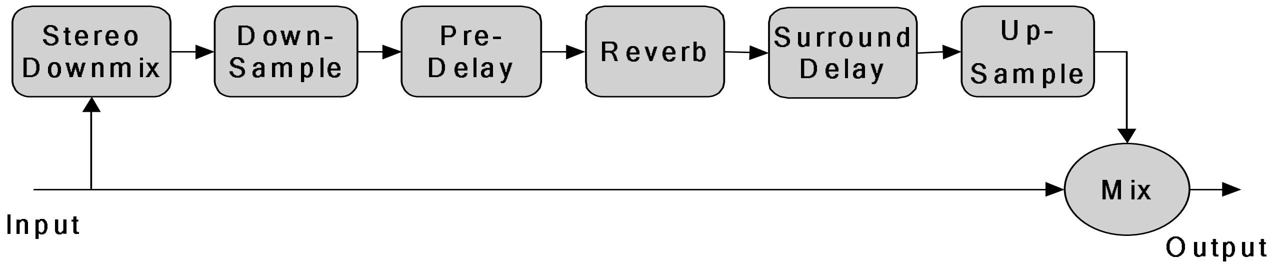
On reading and applying the changed parameters, the RVB1 Algorithm resets the value of the Parameter Change Register to zero.

D.2 Operation

The operation of Room Simulator Number 1 in PA as implemented by the RVB1 Algorithm is described in this section.

D.2.1 Stereo to Quadraphonic Reverberator

FIGURE D-1 **Reverberator Block Diagram**



The reverberator first downmixes its input to stereo. Downmix coefficients are decided by weights modifiable via the “InGain” alpha codes. For example, the Front Input Gain Register (modifiable via writeRVBfrontInGain(N)) controls the weights applied to the front left and right channels.

The resulting stereo signal is then downsampled by a factor of two, passed through a predelay block, and then processed by the reverberator network. The quadraphonic output of the reverberator consists of left, right, left surround and right surround channels. The surround channels are then optionally delayed from the front channels, then all four outputs are upsampled by a factor of two. The outputs are then mixed with the original multi-channel input to form a multi-channel output.

The final mix block uses the Effects Mix Register to determine the wet/dry mix of the final output. The individual mix coefficients applied to the quadraphonic reverberation output can be controlled via the “OutGain” alpha codes. For example, the Front Output Gain Register (modifiable via writeRVBfrontOutGain(N)) controls the weight applied to the front left and right channels.

D.3 Build-time Parameter Definition

The RVB Algorithm has multiple build-time operating configurations. These configurations provide various buffer allocations, which effect RVB operation.

Pre-defined configurations are defined in Table D-2 (RVB Algorithm Configuration Parameters).

TABLE D-2 **RVB Algorithm Configuration Parameters**

Parameter Name	Description
IRVB_PARAMS_STD	In this config RVB allocates a buffer of 24KB in IRAM. This config provides only standard reverb operation and doesn’t provide Pre delay Or Surr delay. RVB can not operate on top of PL2x/NEO/DTS-ES inputs for these parameters.
IRVB_PARAMS_SDELAY	In this config RVB allocates a buffer of 32KB in SDRAM. This config provides standard reverb operation, provides Pre delay, and doesn’t provide Surr Delay. RVB can not operate on top of PL2x/NEO/DTS-ES inputs for these parameters.
IRVB_PARAMS_PSDELAY	In this config RVB allocates a buffer of 40KB in SDRAM. This config provides standard reverb operation, provides Pre delay, and provides Surr delay. RVB can not operate on top of PL2x/NEO/DTS-ES inputs for these parameters. This is the default set of configuration parameters used for all PA builds.
IRVB_PARAMS_PSDELAY_C10	In this config RVB allocates a buffer of 40KB in SDRAM and 1.3KB in IRAM. This config provides standard reverb operation, Pre delay and Surr delay. In addition, with this config RVB can operate on top of PL2x/NEO/DTS-ES inputs.

D.3.1 Output Channel Configuration

The output from the RVB1 Algorithm need not have the same channel configuration as its input. The current implementation supports the combinations given in Table D-3 (Supported Input and Output Channel Configurations).

TABLE D-3 **Supported Input and Output Channel Configurations**

Input/Output	Stereo	3Stereo	Surround1/ Phantom1	Surround2/ Phantom2	Surround3/ Phantom3	Surround4/ Phantom4
Stereo	Supported	Supported	Supported	Supported	Supported	Supported
3Stereo	Supported	Supported	Supported	Supported	Supported	Supported
Surround1/ Phantom1			Supported	Supported	Supported	Supported
Surround2/ Phantom2				Supported	Supported	Supported
Surround3/ Phantom3					Supported[a]	Supported[b]
Surround4/ Phantom4						Supported[c]

[a] Not supported for DTS-ES input on *DA8xx, see below.

[b] *ibid*.

[c] *ibid*.

*DA8xx builds (except for those which use the IRVB_PARAMS_PSDELAY_C10 parameter set) place some restrictions on RVB1 processing in order to more efficiently overlay memory:

- The RVB1 Algorithm will not run if either PL2X or NEO is enabled (via Mode Control or Bypass Control registers) given the current ASP Link Chain ordering in which RVB1 follows both PL2X and NEO. The RVB1 Algorithm is effectively mutually exclusive with these other algorithms. RVB1 can run only if other algorithms using the same Common Memory space are disabled, or if the ASP Link Chain ordering places RVB1 before other algorithms using the same Common Memory space.
- The RVB1 Algorithm will not run with a DTS-ES input (this can be overridden by sending writeDTSOperationalModeDigitalSurround).

There is no damaging effect to leaving RVB1 (Mode or Bypass) enabled in builds; the Common Memory protection algorithm will ensure that RVB1 processing does not take place.

If RVB1 is initialized using the IRVB_PARAMS_PSDELAY_C10 parameter set, the above restrictions are not in effect; in this case RVB1 can operate with any combination of PL2X, NEO, and DTS-ES operation.

D.3.2 Level

Mix, Input Gain, and Output Gain parameters are provided with the RVB algorithm to allow for fine-tuning of the sound. These registers allow for level modifications to be applied to a particular channel or channel pair.

These registers may be changed during audio playback without risk of audio artifacts.

If the level of any of these registers is increased above unity gain (0x4000), clipping may occur. The developer should take care to provide enough headroom in the implementation by utilizing the Master Volume Control or Volume Trim Control Registers to decrease the level such that full range input will not exceed the full range output. For more information on the usage of the Volume Control Usage see Chapter 5.4 (Volume Control)

For the Room Presets, the Audio Size parameter is tuned to provide the amount of headroom required when used in conjunction with the Internal Volume implementation. The setting of the Audio Size parameter varies depending on whether the input is stereo or multichannel and also has some special cases depending on the Room Preset selected. For more details on the settings of the Audio Size parameter see Table D-4 (Audio Size settings).

TABLE D-4 **Audio Size settings**

Input	Worst-case headroom (Center Channel)	
	Presets 0-10, 12-17	Preset 11
Stereo	6 dB	9 dB
Multichannel	12 dB	17 dB

D.3.3 Room Presets

The RVB1 Algorithm provides a total of 18 built-in “preset” rooms to simplify usage. Each preset can be thought of as a predetermined set of values for RVB1 status registers that directly control the reverberator. Table Table D-5 (Room Simulator Number 1 Algorithm Preset Rooms) shows the list of available presets together with their associated parameter set.

TABLE D-5 **Room Simulator Number 1 Algorithm Preset Rooms**

Preset	Name	Size	Mix	LF Cut	Reverb Time	Damping	Pre Delay	Surr Delay	Notes
0	Big Hall	100%	35%	20 Hz	3.5 s	6 kHz	0	0	Default big hall
1	Bright Hall	100%	30%	20 Hz	4.0 s	12 kHz	0	0	Brighter hall
2	Dark Hall	80%	35%	20 Hz	4.0 s	3 kHz	0	0	Darker hall
3	Cathedral	100%	50%	20 Hz	4.0 s	5 kHz	0	0	Big cathedral with lots of surround
4	Club	60%	35%	20 Hz	1.5 s	3 kHz	0	0	Default nightclub
5	Bright Club	70%	40%	20 Hz	1.5 s	12 kHz	0	0	Brighter nightclub
6	Smoky Club	40%	40%	20 Hz	1.5 s	4 kHz	0	0	Smokier nightclub
7	Bright Stadium	100%	40%	20 Hz	5.0 s	10 kHz	0	0	Stadium size, doesn’t corrupt center vocals
8	Arena	100%	40%	20 Hz	5.0 s	4 kHz	0	0	Brighter than “Stadium”
9	Vox Cinema	60%	50%	20 Hz	1.5 s	6 kHz	0	0	For movies with lots of dialog
10	Music Cinema	70%	40%	20 Hz	3.0 s	4 kHz	0	0	For movies with lots of music
11	Bright Space	20%	40%	20 Hz	3.5 s	5 kHz	0	0	Small bright space
12	Bigger Hall	100%	30%	20 Hz	3.0 s	4 kHz	30	30	Bigger hall than “Big Hall”
13	Big Arena	100%	35%	20 Hz	5.0 s	4 kHz	50	40	Bigger arena than “Arena”
14	Biggest Stadium	100%	30%	20 Hz	3.0 s	4 kHz	85	85	Bigger stadium than “Bright Stadium”
15	Jazz Hall	60%	35%	20 Hz	1.5 s	3 kHz	20	30	Jazz hall
16	Slap Happy Club	60%	35%	20 Hz	1.5 s	9 kHz	20	50	Very live sounding nightclub
17	Surround Club	40%	35%	20 Hz	1.5 s	4 kHz	30	30	Live club with lots of surround

D.3.4 Customization Using the Parameters

There are three classes of parameters to consider when designing presets. The static Input Gain parameters can be adjusted to modify the effect for different program material (e.g. Music vs. Dialog). The dynamic (Effects Mix, Reverberation Time, and Damping Frequency) and static parameters (Room Size, Low Frequency Cut, Pre-Delay and Surround Delay) can be used to change the sonic character of the effect. Finally, the static Output Gain parameters may be used to change amount of immersion from the surrounds.

In Figure D-1 (Reverberator Block Diagram), the first processing block is the Stereo Down Mixer. The three Input Gain parameters control the source mix to the reverberator. Nominally, they are all set to unity, but for cinema dialog, one might reduce the Center Input Gain to avoid affecting speech intelligibility. Alternatively, it could be raised for a “voice of God” effect.

The character of the simulated room is set using the three the dynamic parameter controls and the four static parameter controls. The Room Size parameter rescales the entire loop delay structure, thus changing the density and timing of the reflections. The longer the overall loop, the larger the space being modeled. The Reverb Time parameter controls the gain around the loop, simulating the absorption of sound as it bounces off imperfectly reflecting surfaces. The Damping parameter increases the relative decay rates of high versus low frequencies by inserting a Low Pass Filter (LPF) in the loop. The Low Frequency Cut parameter controls the cutoff frequency for a first-order high-pass shelf filter in the send to the Reverb block, effectively removing low frequencies below the filter cutoff from the reverberation signal. The Pre-Delay parameter provides a stereo delay prior to the Reverb block by a specified amount, emulating the distance from the sound source. The Surround Delay parameter provides a stereo delay that delays the surround channels from the front channels simulating greater depth within larger rooms.

There is an optimal correspondence between a room’s size and its reverberation time (depending on what is being performed), but interesting effects can be obtained at the limits. Cinemas are characterized by being fairly dead large rooms. Shower singers are used to relatively live small spaces.

Returning to Figure D-1 (Reverberator Block Diagram), the Mix section is controlled by both the dynamic Effects Mix parameter and the static Output Gain parameters. The Effects Mix parameter is the overall master “wet / dry” control for all channels, and is a prime candidate for front panel adjustment by the end user. The three Output Gain parameters can be individually adjusted to modify the sense of the listener’s location in a room. For example, increasing the Surround Output Gains will increase the sense of being further back in the simulated room.

D.3.5 Usage of Audio Frame Data Structure Registers

A pointer to the Audio Frame Data Structure is passed to an ASP Algorithm as part of both the reset and apply function. It encapsulates a “frame” of audio sample data on which processing is to take place.

Table D-6 (Usage of Audio Frame Data Structure Quantities) illustrates the registers in the Audio Frame Data Structure which are utilized by the RVB1 Algorithm and whether the registers are read, written, or read *and* written.

TABLE D-6 **Usage of Audio Frame Data Structure Quantities**

Audio Frame Data Structure Register	Read/Written
channelConfigurationStream	R/W
channelConfigurationRequest	R
sampleRate	R
sampleCount	R
data.sample	R/W
sampleProcess	W
data.samSize	R/W

The Channel Configuration Stream register is tested for supported channel configuration combinations. If not PAF_CC_SAT_STEREO, PAF_CC_SAT_SURROUND1, PAF_CC_SAT_SURROUND2, PAF_CC_SAT_SURROUND3, or PAF_CC_SAT_SURROUND4, the apply function will exit and will not apply RVB1 processing.

The Channel Configuration Request register is also tested for supported channel configuration combinations. If not PAF_CC_SAT_STEREO, PAF_CC_SAT_SURROUND1, PAF_CC_SAT_SURROUND2, PAF_CC_SAT_SURROUND3, or PAF_CC_SAT_SURROUND4, the apply function will exit and will not apply RVB1 processing.

The algorithm checks for static parameter updates from alpha code changes or from a new preset selection. Additionally, if the incoming Sample Rate registers or Channel Configuration Stream or Request registers change, then the *static* parameters within the algorithm are updated.

The existence of the subwoofer in the Channel Configuration Stream register indicates to the RVB1 Algorithm whether or not to include the LFE in the downmix to the stereo reverberator.

The RVB field of the Sample Process register is set when RVB1 processing has successfully occurred. Additionally, the SURRPROC and BACKPROC fields are set when the output channels are greater than 2 and 5, respectively.

APPENDIX E Matrix

The Matrix Algorithm is an Audio Stream Processing (ASP) Algorithm with an Application Interface (API) that permits one to exercise control over various aspects of its operation. This appendix explains the details of this component of the Performance Audio Framework (PA/F) API.

The Matrix (MTX) Algorithm uses a stereo to multichannel algorithm to expand the sound field of stereo inputs to a multichannel environment. The MTX algorithm includes the capability to fill satellite channels not available in the input.

E.1 Control, Status, and Command Registers

Table E-1 gives a list of alpha code symbols for the MTX Algorithm. For an explanation of the various forms of register usage, see Section 6.1 (Register Architecture).

Please keep the following in mind when reading this table:

- Default settings for control registers are shaded. Default settings result at power-up or reset.
- Alpha code of types `write` and `read` are used to set and get values to and from registers.
- Numeric values for the preprocessor symbols shown in this table and appendix are provided in C-language form in the alpha code symbol file `P:\alpha\mtx_a.h`.

The MTX Algorithm has no command registers.

TABLE E-1 **Matrix Algorithm Alpha Codes**

Register	Alpha Code	Description
Status	readMTXStatus	Return entire MTX Algorithm status structure.
Mode	readMTXMode	Indicate if Matrix Algorithm is enabled or disabled.
	writeMTXModeDisable	Disable the Matrix Algorithm
	writeMTXModeEnable	Enable the Matrix Algorithm
Bypass	readMTXBypass	Return the value of the Bypass Control Register.
	writeMTXBypassEnable	Disable the MTX Algorithm operation.
	writeMTXBypassDisable	Enable the MTX Algorithm operation.
Preset	readMTXPreset	Return the value of the Preset Register.
	writeMTXPreset(N)	Select the Nth preset, where N is a 16 bit value. Valid values for N are -1 and 0-2. The value of -1 has special meaning.
	writeMTXPreset(0)	Set the Preset Register as 0.
	writeMTXPreset(-1)	Set the Preset Register as -1 (0xffff).
	writeMTXMusic	Select the “Music” preset. (Preset room number 0)
	writeMTXDialog	Select the “Dialog” preset. (Preset room number 1)
	writeMTXMultichannelStereo	Select the “MultichannelStereo” preset. (Preset room number 2)
Center Mix	readMTXCtrMix	Return the value of the Center Channel Mix Register.
	writeMTXCtrMix(N)	Set the value of the Center Channel Mix Register to N, a 16 bit value. N is interpreted as the gain amount scaled by 2^{14}.
	writeMTXCtrMix(0x1000)	Set the Center Channel Mix Register to 0.25.
Sub Mix	readMTXSubMix	Return the value of the Subwoofer Channel Mix Register.
	writeMTXSubMix(N)	Set the value of the Subwoofer Channel Mix Register to N, a 16 bit value. N is interpreted as the gain amount scaled by 2^{14}.
	writeMTXSubMix(0x4000)	Set the Subwoofer Channel Mix Register to 1.
Surround Dry Mix	readMTXSurrDry	Return the value of the Surround Channel Dry Mix Register.
	writeMTXSurrDry(N)	Set the value of the Surround Channel Dry Mix Register to N, a 16 bit value. N is interpreted as the gain amount scaled by 2^{14}.
	writeMTXSurrDry(0x0000)	Set the Surround Channel Dry Mix Register to 0.
Surround Wet Mix	readMTXSurrWet	Return the value of the Surround Channel Wet Mix Register.
	writeMTXSurrWet(N)	Set the value of the Surround Channel Wet Mix Register to N, a 16 bit value. N is interpreted as the gain amount scaled by 2^{14}.
	writeMTXSurrWet(0x4000)	Set the Surround Channel Wet Mix Register to 1.
Back Dry Mix	readMTXBackDry	Return the value of the Back Channel Dry Mix Register.
	writeMTXBackDry(N)	Set the value of the Back Channel Dry Mix Register to N, a 16 bit value. N is interpreted as the gain amount scaled by 2^{14}.
	writeMTXBackDry(0x0000)	Set the Back Channel Dry Mix Register to 0.

Register	Alpha Code	Description
Back Wet Mix	readMTXBackWet	Return the value of the Back Channel Wet Mix Register.
	writeMTXBackWet(N)	Set the value of the Back Channel Wet Mix Register to N, a 16 bit value. N is interpreted as the gain amount scaled by 2^{14}.
	writeMTXBackWet(0x4000)	Set the Back Channel Wet Mix Register to 1.
Surround Gain	readMTXSurrGain	Return the value of the Surround Channel Gain Register.
	writeMTXSurrGain(N)	Set the value of the Surround Channel Gain Register to N, a 16 bit value. N is interpreted as the gain amount scaled by 2^{14}.
	writeMTXSurroundGain(0x3000)	Set the Surround Channel Gain Register to 0.75.
Back Gain	readMTXBackGain	Return the value of the Back Channel Gain Register.
	writeMTXBackGain(N)	Set the value of the Back Channel Gain Register to N, a 16 bit value. N is interpreted as the gain amount scaled by 2^{14}.
	writeMTXBackGain(0x4000)	Set the Back Channel Gain Register to 1.
Surround Delay	writeMTXsurDelay	Return the value of the Surround Delay Register.
	writeMTXsurDelay(N)	Set the value of the Surround Delay Register to N, a 16 bit value. Valid values are 0-42 msec.
	writeMTXsurDelay(42)	Set the Surround Delay Register to 42 msec.

E.1.1 Mode

The MTX Mode Control Register controls the basic operation of the MTX Algorithm. If zero, operation is disabled, and if non-zero, operation is enabled so long as operation is not bypassed as per the next section).

E.1.2 Bypass Control

The MTX Bypass Control Register provides the ability to enable and disable the MTX Algorithm.

If the Bypass Control Register is set using writeMTXBypassEnable then the MTX Algorithm *is not* applied to the input.

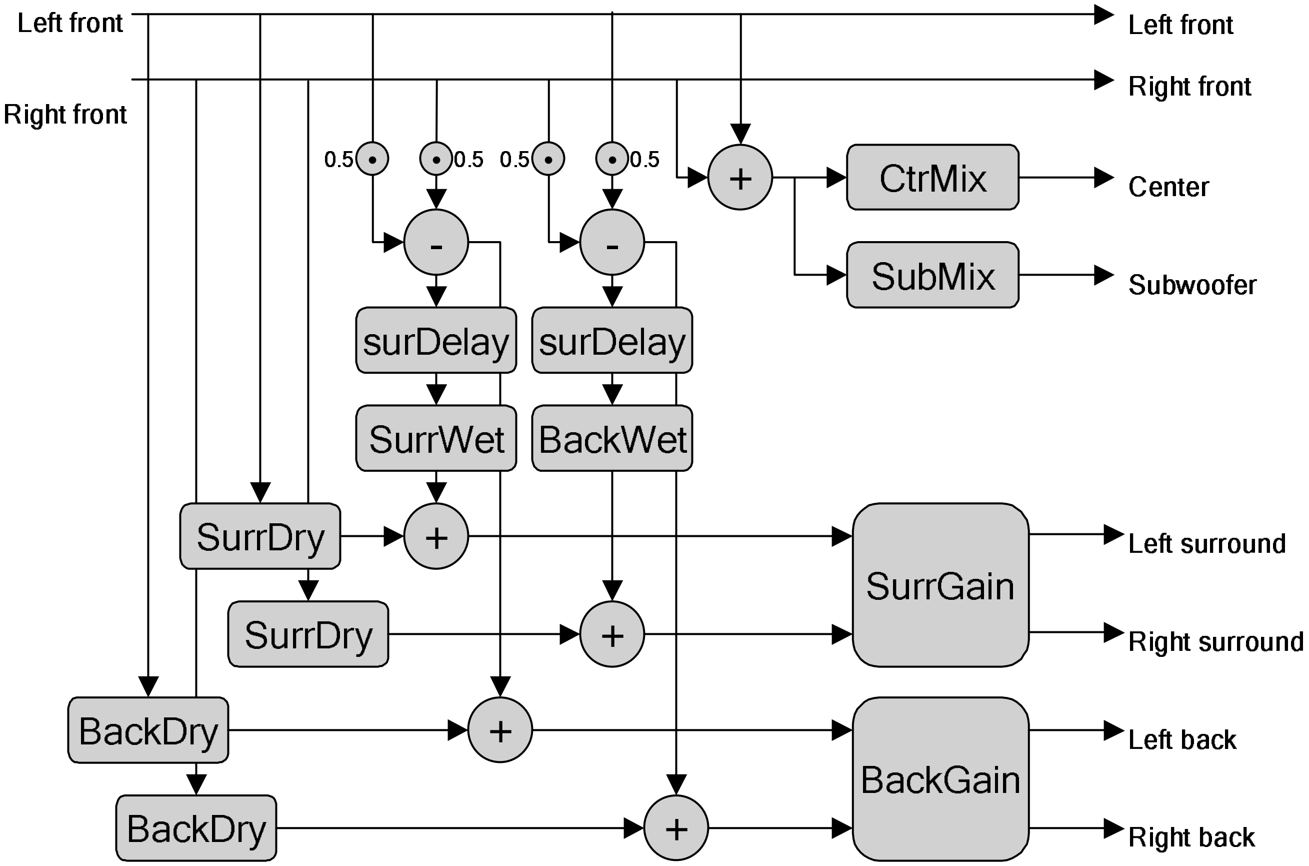
If the Bypass Control Register is set using writeMTXBypassDisable then the MTX Algorithm *is* applied to the input.

E.2 Operation

The operation of MTX in PA as implemented by the MTX Algorithm is described in this section.

E.2.1 Matrix Algorithm

FIGURE E-1 **Matrix Block Diagram**



The Matrix Algorithm derives multichannel output from stereo input, with a number of parameters to tailor the sound to varying preferences.

The output of the channels derived from the stereo input can be scaled with the Center Channel Mix, Subwoofer Channel Mix, Surround Channel Gain, and Back Channel Gain registers.

The surround and back channels may be blended with either the direct, scaled input from the stereo input channels or with a delayed version of the difference between the input channels. The direct, scaled input is controlled by the Surround Channel Dry Mix and Back Channel Dry Mix registers. The delayed difference input is controlled by the Surround Channel Wet Mix and Back Channel Wet Mix registers. Additionally, the delay may be modified to be in the range of 0 - 42 milliseconds using the Surround Delay register.

E.2.2 Output Channel Configuration

The output from the MTX Algorithm need not have the same channel configuration as its input. The current implementation supports the combinations given in Table E-2 (Supported Input and Output Channel Configurations).

TABLE E-2 **Supported Input and Output Channel Configurations**

Output	Input	
	Stereo	Any Other
Surround0	Supported	Not Supported
Phantom1	Supported	Not Supported
Surround1	Supported	Not Supported
Phantom2	Supported	Not Supported
Surround2	Supported	Not Supported
Phantom3	Supported	Not Supported
Surround3	Supported	Not Supported
Phantom4	Supported	Not Supported
Surround4	Supported	Not Supported

If MTX operates, it additionally produces a Subwoofer output if the channel configuration requested includes subwoofer output. Note that since MTX does not operate when Stereo or Phantom0 output is requested, MTX will not produce a Subwoofer output in these cases, even if Stereo_1 is requested.

E.2.3 Level

Mix, Wet, Dry, and Gain parameters are provided with the MTX algorithm to allow for fine-tuning of the sound. These registers allow for level modifications to be applied to a particular channel or channel pair.

These registers may be changed during audio playback without risk of audio artifacts.

If the level of any of these registers is increased above unity gain (0x4000), clipping may occur. The developer should take care to provide enough headroom in the implementation by utilizing the Master Volume Control or Volume Trim Control Registers to decrease the level such that full range input will not exceed the full range output. For more information on the usage of the Volume Control Usage see Chapter 5.4 (Volume Control)

E.2.4 Matrix Presets

The user may directly control individually the nine setting registers which affect the MTX Algorithm's output:

- CtrMix/SubMix

- SurrWet/SurrDry
- SurrGain/BackGain
- BackDry/BackWet
- surDelay

Additionally or alternatively, the user may select one of three “preset” combinations of the nine settings by entering a value 0, 1, or 2 into the Preset Register.

When the MTX Algorithm operates, it recognizes the non-default (i.e., non-0xffff) value of the Preset Register, loads the nine setting registers with the appropriate preset values (see Table E-3 (Matrix Algorithm Presets)), and then sets the Preset Register back to the default value (-1/0xffff).

Note that if the Preset Register is set to a value other than the default (-1/0xffff), the Preset register will remain at that value until either it is set back to 0xffff (effectively cancelling the request), or the MTX Algorithm operates (in which case it sets the nine registers appropriately).

Some situations wherein the Preset Register would remain at a setting other than default (0xffff) indefinitely include:

- The MTX Bypass Register is set to “BypsssEnable” (`writeMTXBypassEnable`).
- The output of the decoder is not stereo.
- Another Surround Processing ASP Algorithm (e.g., PL2X or NEO) precede the MTX Algorithm and converts stereo to multichannel.
- The ASP Chain is disabled (e.g. via `writeDECASPGearControlNil`).

If an invalid Preset Register value is written (e.g., a value other than -1, 0, 1, or 2), then the following will occur:

- The Preset Register will have no effect.
- The invalid value will remain in the Preset Register until a valid Preset value (-1, 0, 1, or 2) is written

When a preset is selected, MTX will set all nine setting registers according to that preset. If any of the setting registers need to be further modified, it should be done after MTX acts on the preset and loads the nine registers. So, overriding any value of preset will not work from cus_atboot or from the micro if the overriding commands are sent without confirming that the preset has been processed. MTX must get a chance to execute and load the preset before you override any value.

So, the recommendation is to choose one of following 3 ways:

1. Use only presets

2. If overriden is needed, set all individual values rather than preset.

3. After writing to the preset register, check one of the nine setting registers to know when it gets updated and then override any setting.

TABLE E-3 **Matrix Algorithm Presets**

Preset	Name	Cntr Mix	Sub Mix	Surr Dry	Surr Wet	Back Dry	Back Wet	Surr Gain	Back Gain	Surr Del	Notes
-1	Default	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	No settings are modified
0	Music	0	0.31	0	1	0	1	1.25	1	42	Music
1	Dialog	0.5	0.31	0	1	0	1	0.75	1	42	Dialog - movies
2	Multichannel Stereo	0.5	0.16	1	0	1	0	0.71	0.35	0	Multichannel stereo

E.2.5 Usage of Audio Frame Data Structure Registers

A pointer to the Audio Frame Data Structure is passed to an ASP Algorithm as part of both the reset and apply function. It encapsulates a “frame” of audio sample data on which processing is to take place.

Table E-4 (Usage of Audio Frame Data Structure Quantities) illustrates the registers in the Audio Frame Data Structure which are utilized by the MTX Algorithm and whether the registers are read, written, or read *and* written.

TABLE E-4 **Usage of Audio Frame Data Structure Quantities**

Audio Frame Data Structure Register	Read/Written
channelConfigurationStream	R/W
channelConfigurationRequest	R
sampleRate	R
sampleCount	R
data.sample	R/W
sampleProcess	R/W

The Channel Configuration Stream register is tested for supported channel configuration combinations. If not PAF_CC_SAT_STEREO the apply function will exit and will not apply RVB1 processing. Additionally, the Channel Configuration Stream register is updated to reflect the expansion of channels resulting from MTX processing.

The Channel Configuration Request register is also tested for supported channel configuration combinations. If not PAF_CC_SAT_STEREO, PAF_CC_SAT_SURROUND1, PAF_CC_SAT_SURROUND2, PAF_CC_SAT_SURROUND3, or PAF_CC_SAT_SURROUND4, the apply function will exit and will not apply MTX processing.

The algorithm checks for changes in certain parameters from frame to frame. If the incoming Sample Rate register or Surround Delay register change, then the *static* parameters within the algorithm are updated. In the case of the Sample Rate register, the supported rates are 32 kHz, 44.1 kHz, and 48 kHz. If rates other than the supported rates are input to the MTX Algorithm, processing will continue, but the Surround Delay length will be approximated using 48 kHz.

The MTX field of the Sample Process register is set when MTX processing has successfully occurred. Additionally, the SURRPROC and BACKPROC fields are set when the output channels are greater than 2 and 5, respectively.

APPENDIX F Downmix Algorithm Number 1

The Downmix (DMX) Algorithm is an Audio Stream Processing (ASP) Algorithm with an Application Interface (API) that permits one to exercise control over various aspects of its operation. This appendix explains the details of this component of the Performance Audio Framework (PA/F) API.

Downmix Algorithm Number 1 provides a limited but powerful form of downmix suitable for use with PA. In particular, it provides mixing capability that mimics that of the PCM Algorithm in many respects. Its primary use is as an adjunct to custom surround processing ASP Algorithms that do not themselves provide full downmix functionality as required.

F.1 Control, Status, and Command Registers

Table F-1 gives a list of alpha code symbols for the DMX Algorithm. For an explanation of the various forms of register usage, see Section 6.1 (Register Architecture).

Please keep the following in mind when reading this table:

- Default settings for control registers are shaded. Default settings result at power-up or reset.
- Bit-mapped registers allow the specification of multiple values simultaneously using the logical-or of any combination of the command or response for that register.
- Alpha code of types `write` and `read` are used to set and get values to and from registers, respectively, and alpha code of type `exec` is used to cause actions to occur.
- Numeric values for the preprocessor symbols shown in this table and appendix are provided in C-language form in the alpha code symbol file `P:\alpha\dmx_a.h`.

The DMX Algorithm has no command registers.

TABLE F-1 **Downmix Alpha Code**

Register	Alpha Code	Description
Status	readDMXStatus	Return entire DMX status structure.
Mode	readDMXMode	Return DMX Mode Control Register value.
	writeDMXModeDisable	Disable Downmix.
	writeDMXModeEnable	Enable Downmix.
Program Spec	readDMXProgramSpec	Return DMX Program Spec Control Register value.
	writeDMXProgramSpecAuto	Select automatic determination of Program Channel Configuration to optimize execution efficiency (minimum MIPS).
	writeDMXProgramSpecUnknown	Select the indicated Program Channel Configuration for demonstration or testing purposes as described in Section F.1.2 and in Section F.2.2.
	writeDMXProgramSpecNone	
	writeDMXProgramSpecMono	
	writeDMXProgramSpecStereo	
	writeDMXProgramSpecSurround2	
	writeDMXProgramSpecSurround4	
Program Type	readDMXProgramType	Provide information about the program interpretation as described in Section F.1.3.
	wroteDMXProgramTypeUnknown	The Program Channel Configuration is unknown.[a]
	wroteDMXProgramTypeStream	The Program Channel Configuration does exactly match the Stream Channel Configuration.
	wroteDMXProgramTypeStreamNot	The Program Channel Configuration does not exactly match the Stream Channel Configuration.
Downmix Type	readDMXDownmixType	Provide information about the downmix operation as described in Section F.1.4.
	wroteDMXDownmixTypeUnknown	The Downmix Channel Configuration is unknown.[b]
	wroteDMXDownmixTypeNone	The Downmix Channel Configuration does exactly match the Stream Channel Configuration, so no downmix is performed.
	wroteDMXDownmixTypeRequest	The Downmix Channel Configuration does not exactly match the Stream Channel Configuration but does exactly match the Request Channel Configuration, and downmix is performed.
	wroteDMXDownmixTypeRequest-Not	The Downmix Channel Configuration does not exactly match the Stream Channel Configuration and does not exactly match the Request Channel Configuration, but downmix is performed.
LFE Downmix Include	readDMXLFEDownmixInclude	Return DMX LFE Downmix Include Control Register value.
	writeDMXLFEDownmixInclude-No	Do not include LFE Channel in downmix.
	writeDMXLFEDownmixInclude-Yes	Do include LFE Channel in downmix.

Register	Alpha Code	Description
LFE Downmix Volume	readDMXLFEDownmixVolume	Return DMX LFE Downmix Volume Control Register value.
	writeDMXLFEDownmix-VolumeN(2*10)	Set DMX LFE Downmix Volume Control Register to 10 dB.
	writeDMXLFEDownmix-VolumeN(*NN*)	Set DMX LFE Downmix Volume Control Register to *NN* dB in units of 0.5 dB.
Request Channel Configuration	readDMXChannelConfigurationRequest	Return DMX Request Channel Configuration Control Register value.
	writeDMXChannelConfiguration-RequestStereoUnknown	Downmix DMX input as described in Section F.1.7, according to the Audio Stream Channel Configuration Request Control Register.
	writeDMXChannelConfiguration-Request*X*	Downmix DMX input as described in Section F.1.7, according to the PA/F-standard channel configuration value *X* as given in Table 4-3 (Alpha Code for Channel Configuration Registers).
Program Channel Configuration	readDMXChannelConfiguration-Program	Indicate the manner in which the DMX input is interpreted as described in Section F.1.8.
	wroteDMXChannelConfiguration-Program*X*	The numerical values returned are PA/F-standard channel configuration values as given in Table 4-3 (Alpha Code for Channel Configuration Registers).
Downmix Channel Configuration	readDMXChannelConfiguration-Downmix	Indicate the manner in which the DMX output has been produced as described in Section F.1.9.
	wroteDMXChannelConfiguration-Downmix*X*	The numerical values returned are PA/F-standard channel configuration values as given in Table 4-3 (Alpha Code for Channel Configuration Registers).

[a] This value results following ASP Reset and preceding ASP Application.

[b] *ibid*.

F.1.1 Mode

The DMX Mode Control Register controls basic operation of the DMX Algorithm. If zero, operation is disabled, and if non-zero, operation is enabled.

F.1.2 Program Spec

The DMX Program Spec Control Register can be used to override the automatic determination of the manner in which DMX input is processed as described in Section F.2.2. The default setting is appropriate for automatic determination during normal system operation.

The inclusion or exclusion of the LFE Channel in the downmix is controlled as outlined in Section F.1.5. The interpretation of the DMX Program Spec Control Register is subject to this selection.

F.1.3 Program Type

The DMX Program Type Status Register gives extra information regarding the interpretation of the DMX input that is useful mainly during test and debug:

- If the program type is “unknown,” the ASP Algorithm has been reset but not applied to any sample data.

- If the program type is “stream,” the ASP Algorithm has been applied to sample data, and the channel configuration of the DMX input (Audio Stream Channel Configuration Stream) does match the channel configuration used in DMX downmix processing (DMX Channel Configuration Program).
- If the program type is “not stream,” the ASP Algorithm has been applied to sample data, but the channel configuration of the DMX input (Audio Stream Channel Configuration Stream) does not match the channel configuration used in DMX downmix processing (DMX Channel Configuration Program).

  This fact is not particularly pertinent except during test and debug.

The definition of the term “match” above is subject to the inclusion or exclusion of the LFE Channel in the downmix as outlined in Section F.1.5. If the LFE Channel is excluded from the downmix, as is the default condition, then the “match” is determined while ignoring the subwoofer portion of the channel configurations. If the LFE Channel is included in the downmix, then no special handling here is required.

The definition of the term “match” above does not include a comparison of the value of the auxiliary field of the channel configurations.

F.1.4 Downmix Type

The DMX Downmix Type Status Register gives extra information regarding operation of DMX downmix that is useful mainly during test and debug:

- If the downmix type is “unknown,” the ASP Algorithm has been reset but not applied to any sample data.
- If the downmix type is “none,” the ASP Algorithm has operated on sample data. However, the channel configuration of the DMX input (Audio Stream Channel Configuration Stream) does match the channel configuration requested in DMX downmix processing (DMX Channel Configuration Downmix), and so no actual operation on the audio data is needed.

  This indicates that no actual downmix is needed or has taken place.
- If the downmix type “is request,” the ASP Algorithm has been applied to sample data, and the channel configuration of the DMX output (DMX Channel Configuration Downmix) does match the channel configuration requested.

  This indicates that exactly the requested downmix has taken place.
- If the downmix type “is not request,” the ASP Algorithm has been applied to sample data, but the channel configuration of the DMX output (DMX Channel Configuration Downmix) does not match the channel configuration requested.

  This indicates that downmix, but not exactly the requested downmix, has taken place.

The definition of the term “match” above is subject to the inclusion or exclusion of the LFE Channel in the downmix as outlined in Section F.1.5. If the LFE Channel is excluded from the downmix, as is the default condition, then the “match” is determined while ignoring the subwoofer portion of the channel configurations. If the LFE Channel is included in the downmix, then no special handling here is required.

The definition of the term “match” above does not include a comparison of the value of the auxiliary field of the channel configurations.

F.1.5 LFE Downmix Include

The DMX LFE Downmix Include Control Register determines whether material from the LFE Channel is included in the downmix. By default, such material is not included. Rather, it is assumed that Bass Management will be used for this purpose. An alternative setting of this control register allows inclusion of the LFE Channel in the downmix.

If the LFE Channel is not included in the downmix, it is passed from the input of the DMX Algorithm to the output without modificaition. The channel configuration control and status registers of both the DMX Algorithm and the Audio Stream reflect this passing.

The setting of this control register has implications for how other control registers are interpreted and how the value of status registers is produced:

- The interpretation of the DMX Program Spec Register is subject to this setting. See Section F.1.2.
- The generation of the DMX Program and Downmix Type Status Register values are subject to this setting. See Section F.1.3 and Section F.1.4, respectively.
- The interpretation of the DMX Channel Configuration Request Control Register is subject to this setting. See Section F.1.7.
- The generation of the DMX Channel Configuration Program and Downmix Status Register values are subject to this setting. See Section F.1.8 and Section F.1.9, respectively.

F.1.6 LFE Downmix Volume

The DMX LFE Downmix Volume Control Register is used to set a gain to be applied to the LFE Channel of DMX input as part of downmix. It is given in units of 0.5 dB, so that the value 2*10 or 0x14 represents 10 dB. The default value of the DMX LFE Downmix Volume Control Register is 10 dB.

Two important caveats apply regarding application of the gain specified by the DMX LFE Downmix Volume Control Register:

- This gain is applied only as part of downmix that involves the LFE Channel. It is not applied to the LFE Channel at the DMX input in the general case.
- The audio size generated by downmix is set as per the default 10 dB value of this register. Changing the gain does not change the audio size produced as a result of downmix. A gain greater than 10 dB or less than 0 dB is not recommended.

F.1.7 Channel Configuration Request

The DMX Channel Configuration Request Control Register controls the form of downmix:

- The default value “unknown” causes the Audio Stream Channel Configuration Request to be used to control the form of downmix.

  This value is appropriate for use under normal operating conditions.

- Other values cause the DMX Channel Configuration Request to be used to control the form of downmix.

  The Surround selections are subject to the inclusion or exclusion of the LFE Channel in the downmix as outlined in Section F.1.5. The other selections are unaffected by this setting.

  This value is appropriate for use during test and debug.

F.1.8 Channel Configuration Program

The DMX Program Channel Configuration Status Register indicates the overall manner in which DMX input is interpreted for purposes of downmix. If this channel configuration does not match that of the data in the audio stream at the DMX input, the data in the audio stream is padded out to achieve the form needed for implementation of downmix.

The auxiliary field of this register does not reflect that of the Audio Stream. This lack does not affect downmix, as the Audio Stream Channel Configuration rather than this register value is utilized as appropriate for selection of appropriate downmix operations.

The value of this register is not particularly pertinent except during test and debug.

F.1.9 Channel Configuration Downmix

The DMX Downmix Channel Configuration Status Register indicates the manner in which DMX output has been produced.

The value of this register is not particularly pertinent except during test and debug.

F.2 Operation

The operation of Downmix Algorithm Number 1 is described in this section.

F.2.1 Default Operation

With default register settings, the DMX Algorithm performs downmix on the audio stream as requested, automatically, as part of the normal operation of Audio Stream Processing. This default operation is a first, best fit method, subject to the following:

- The standard placement of the DMX Algorithm is just preceding Bass Management. Its operation thus follows all surround processing but precedes bass processing.
- Downmix occurs as requested by the audio stream, not as requested local to the algorithm.
- The LFE Channel is excluded from the downmix. The contents of this channel are preserved for use by later Audio Stream Processing Algorithms, specifically, the Bass Management Algorithm.

The downmix is performed as described in Section F.2.4.

F.2.2 Test Operation

With register settings other than the default, the DMX Algorithm performs downmix on the audio stream as requested, explicitly, as part of demonstration, test, or debug operations. The downmix is performed as described in Section F.2.4. The special control is described below.

As described in Section F.1.5, if the DMX LFE Downmix Include Control Register is set to the value “yes,” the LFE Channel is included in the downmix. By default, it is not included in the downmix but rather preserved as a separate channel for handling by subsequent Audio Stream Processing Algorithms, specifically, the Bass Management Algorithm.

As described in Section F.1.2, if the DMX Program Spec Control Register is set to a value other than “auto,” this value overrides the selection of a downmix routine that would otherwise be performed automatically as part of the normal operation of Audio Stream Processing as described in Section F.2.1. Such a setting can be used as part of special operations to explore the operation of the DMX Algorithm, but such is not typically used during normal system operation.

As described in Section F.1.7, if the DMX Channel Configuration Request Control Register is set to a value other than “unknown,” this value overrides the downmix request that would otherwise be obtained automatically as part of the normal operation of Audio Stream Processing as described in Section F.2.1. Such a setting can be used as part of special operations to explore the operation of the DMX Algorithm, but such is not typically used during normal system operation.

F.2.3 Alternate Operation

With register settings other than the default, the DMX Algorithm performs downmix on the audio stream as requested, implicitly or explicitly, as part of alternate forms of operation. The downmix is performed as described in Section F.2.4. The special control is described below.

As described in Section F.1.5, if the DMX LFE Downmix Include Control Register is set to the value “yes,” the LFE Channel is included in the downmix. This setting can be used when Bass Management is not to be subsequently applied to the audio stream.

F.2.4 Downmix

Downmix operates in a manner equivalent to that of the PCM Algorithm as described in Chapter D.2 (Downmix).

F.2.5 Usage of Audio Frame Data Structure Registers

A pointer to the Audio Frame Data Structure is passed to an ASP Algorithm as part of both the reset and apply function. It encapsulates a “frame” of audio sample data on which processing is to take place.

Table F-2 (Usage of Audio Frame Data Structure Quantities) illustrates the registers in the Audio Frame Data Structure which are utilized by the DMX1 algorithm and whether the registers are read, written, or read *and* written.

TABLE F-2 **Usage of Audio Frame Data Structure Quantities**

Audio Frame Data Structure Register	Read/Written
channelConfigurationStream	R/W
channelConfigurationRequest	R
sampleCount	R
data.sample	R/W
sampleProcess	R/W
data.samSize	R/W

The Channel Configuration Stream registers are used to determine the downmix request. As described in Section F.1.7 (Channel Configuration Request), if the downmix request via the DMX Request Channel Configuration register is any value other than PAF_CC_UNKNOWN, the DMX Request Channel Configuration register determines the downmix. If the DMX Request Channel Configuration is PAF_CC_UKNOWN, the Audio Stream Channel Configuration Request register determines the downmix request used by the DMX1 algorithm. There are channel configurations not supported by DMX1 via the DMX Request Channel Configuration register which are indicated in Table F-3 (Supported Channel Configurations by DMX1).

TABLE F-3 **Supported Channel Configurations by DMX1**

Channel Configuration	DMX Request Channel Configuration Support?
PAF_CC_UNKNOWN	Yes (signals to use Audio Stream Channel Configuration request)
PAF_CC_NONE	Yes
PAF_CC_STEREO	Yes
PAF_CC_PHANTOM1	No
PAF_CC_PHANTOM2	No
PAF_CC_PHANTOM3	No
PAF_CC_PHANTOM4	No
PAF_CC_3STEREO	No
PAF_CC_SURROUND1	No
PAF_CC_SURROUND2	Yes
PAF_CC_SURROUND3	No
PAF_CC_SURROUND4	Yes

The algorithm checks to see if there are any channels requested from the Audio Stream Channel Configuration Request register but not present in the Audio Stream Channel Configuration Program register. For such channels that are requested, but not present in the program, their data and Sample Size are cleared prior to actual downmix processing.

During the downmix processing, the Sample Size register in the Audio Frame Data Data Structure is updated based on the headroom required for the number of channels being downmixed.

The DMX field of the Sample Process register is set when DMX1 downmix processing has successfully occured. The DMX field is not set when DMX1 is enabled but no downmix processing occurs.

APPENDIX G Graphic Equalizer, Number 3

The Graphic Equalizer Algorithm, Number 3, is an Audio Stream Processing (ASP) Algorithm with an Application Interface (API) that permits one to exercise control over various aspects of its operation. This appendix explains the details of this component of the Performance Audio Framework (PA/F) API.

The Graphic Equalizer Algorithm, Number 3 (GEQ3) may be used in several ways.

- As a 2- to 12-band graphic equalizer (band levels change at runtime)
- As a 2- to 12-band parametric equalizer (band frequencies, Q, and levels change at runtime)
- As a loudness compensation filter

Additionally, PA SDK can choose to enable individual channel parameters. It is also possible to enable high-precision filters for one or more equalizer bands in systems where having a very low noise floor is important. In all cases, GEQ3 attempts to handle changes to the level setting of individual bands in a manner that avoids audio artifacts.

In order to minimise the amount of memory used by the GEQ3 Algorithm in the system, several different parameter sets, each supporting a different combination of maximum bands supported, channel-wise coefficients and high-precision filtering are made available for users of the PA SDK to choose from. It is also possible for an SDK user to design a completely new parameter set as per their product requirements.

In its default instantiation, the GEQ3 Algorithm allows for control over the level, width and center frequency of each of up to 10 bands that operate across up-to 8 channels of audio data.

When instantiated as a **loudness compensation** filter (see Appendix G.1.8), GEQ3's bands, frequencies, and levels cannot be adjusted. Instead, loudness can either be enabled or disabled.

When instantiated with "**ganged coefficients**" (see Appendix G.1.9), the same set of coefficients apply to all filtered channels.

When instantiated with "**channel-wise coefficients**" (see Appendix G.1.9), each channel's filtering is individually configurable.

G.1 Control, Status, and Command Registers

Tables N-1, N-2, and N-3 give lists of alpha code symbols for the GEQ3 Algorithm. For an explanation of the various forms of register usage, see Section 6.1 (Register Architecture).

Please keep the following in mind when reading these tables:

- Default settings for control registers are shaded. Default settings result at power-up or reset.
- Alpha code of types `write` and `read` are used to set and get values to and from registers.
- Numeric values for the preprocessor symbols shown in these tables and appendix are provided in C-language form in the alpha code symbol file `P:\alpha\geq3_a.h`.

The GEQ3 Algorithm has no command registers.

Alpha code depends upon whether GEQ3 is instantiated for loudness compensation (Table G-1), ganged coefficients (Table G-2), or channel-wise coefficients (Table G-3). In either of the latter two cases, GEQ3 may be used with different parameter sets (See Section G.1.9) to realize a 2- to a 12-band equalizer. In all cases, only those registers that are relevant to the chosen parameter set may be read or written as the case may be. For example, settings for Band 4 are invalid when using a GEQ3 as a 3-band equalizer. Conversely, settings for Band 11 and 12 are valid when a custom parameter set with the maximum number of bands sets to 12 is used.

TABLE G-1 **Graphic Equalizer Alpha Code (Loudness Compensation)**

Register	Alpha Code	Description
Control	readGEQLOUControl *or* readLOUControl	Return entire LOU status structure.
Loudness Compensation Mode	readGEQLOUMode *or* readLOUMode	Indicate if Loudness Compensation is enabled or disabled.
	writeGEQLOUModeDisable *or* writeLOUModeDisable	Disable Loudness Compensation.
	writeGEQLOUModeEnable *or* writeLOUModeEnable	Enable Loudness Compensation.
Loudness Compensation Bypass	readGEQLOUBypass *or* writeLOUBypass	Return the value of the Bypass Control Register.
	writeGEQLOUBypassEnable *or* writeLOUBypassEnable	Disable the LOU Algorithm operation.
	writeGEQLOUBypassDisable *or* writeLOUBypassDisable	Enable the LOU Algorithm operation.

TABLE G-2 Graphic Equalizer Alpha Code (Ganged coefficients)

Register	Alpha Code	Description
Mode	readGEQMode	Indicate if Graphic Equalizer is enabled or disabled.
	writeGEQModeDisable	Disable Graphic Equalizer.
	writeGEQModeEnable	Enable Graphic Equalizer.
Use	readGEQUse	Return the value of the Use Control Register.
	writeGEQUseDisable	Disable the Use Control Register
	writeGEQUseEnable	Enable the Use Control Register
Parameter Change	readGEQparamChg	Return the current value of the parameter change register.
	writeGEQparamChgRamp	Signal a change to parameters and require a ramped change to coefficients.
	writeGEQparamChgNoRamp	Signal a change to parameters without requiring a ramped change to coefficients.
Number of Bands	readGEQnumBands	Return number of active equalizer bands.
	writeGEQnumBands(N)	Set the number of active bands to N, where N is a decimal value, inclusive. Only values between 2 and the lower of 12 and the value of “Max Bands” as shown in Table G-6 are valid.
	writeGEQnumBands(10)	Set the number of active bands to 10.
Band 1 Center Frequency	readGEQ1Freq	Return center frequency (in Hertz) of the first equalizer band.
	writeGEQ1Freq(N)	Set the center frequency for the first equalizer band to N Hz. N is a 16 bit value.
	writeGEQ1Freq(32)	Set the center frequency for the first equalizer band to 32 Hz.
Band 1 Width	readGEQ1Width	Return the width (in tenths of octaves) of the first equalizer band.
	writeGEQ1Width(N)	Set the width of the first equalizer band to N/10 octaves. N is a 16 bit value.
	writeGEQ1Width(20)	Set the width of the first equalizer band to 2 octaves.
Band 1 Level	readGEQ1Level	Return the level setting (in tenths of dB) of the first equalizer band.
	writeGEQ1Level(N)	Set the level of the first equalizer band to N/10 dB. N is a 16 bit value.
	writeGEQ1Level(0)	Set the level of the first equalizer band to 0 dB.
Band 2 Center Frequency	readGEQ2Freq	Return center frequency (in Hertz) of the second equalizer band.
	writeGEQ2Freq(N)	Set the center frequency for the second equalizer band to N Hz. N is a 16 bit value.
	writeGEQ2Freq(63)	Set the center frequency for the second equalizer band to 63 Hz.
Band 2 Width	readGEQ2Width	Return the width (in tenths of octaves) of the second equalizer band.
	writeGEQ2Width(N)	Set the width of the second equalizer band to N/10 octaves. N is a 16 bit value.
	writeGEQ2Width(20)	Set the width of the second equalizer band to 2 octaves.

Register	Alpha Code	Description
Band 2 Level	readGEQ2Level	Return the level setting (in tenths of dB) of the second equalizer band.
	writeGEQ2Level(N)	Set the level of the second equalizer band to N/10 dB. N is a 16 bit value.
	writeGEQ2Level(0)	Set the level of the second equalizer band to 0 dB.
Band 3 Center Frequency	readGEQ3Freq	Return center frequency (in Hertz) of the third equalizer band.
	writeGEQ3Freq(N)	Set the center frequency for the third equalizer band to N Hz. N is a 16 bit value.
	writeGEQ3Freq(125)	Set the center frequency for the third equalizer band to 125 Hz.
Band 3 Width	readGEQ3Width	Return the width (in tenths of octaves) of the third equalizer band.
	writeGEQ3Width(N)	Set the width of the third equalizer band to N/10 octaves. N is a 16 bit value.
	writeGEQ3Width(20)	Set the width of the third equalizer band to 2 octaves.
Band 3 Level	readGEQ3Level	Return the level setting (in tenths of dB) of the third equalizer band.
	writeGEQ3Level(N)	Set the level of the third equalizer band to N/10 dB. N is a 16 bit value.
	writeGEQ3Level(0)	Set the level of the third equalizer band to 0 dB.
Band 4 Center Frequency	readGEQ4Freq	Return center frequency (in Hertz) of the fourth equalizer band.
	writeGEQ4Freq(N)	Set the center frequency for the fourth equalizer band to N Hz. N is a 16 bit value.
	writeGEQ4Freq(250)	Set the center frequency for the fourth equalizer band to 250 Hz.
Band 4 Width	readGEQ4Width	Return the width (in tenths of octaves) of the fourth equalizer band.
	writeGEQ4Width(N)	Set the width of the fourth equalizer band to N/10 octaves. N is a 16 bit value.
	writeGEQ4Width(20)	Set the width of the fourth equalizer band to 2 octaves.
Band 4 Level	readGEQ4Level	Return the level setting (in tenths of dB) of the fourth equalizer band.
	writeGEQ4Level(N)	Set the level of the fourth equalizer band to N/10 dB. N is a 16 bit value.
	writeGEQ4Level(0)	Set the level of the fourth equalizer band to 0 dB.
Band 5 Center Frequency	readGEQ5Freq	Return center frequency (in Hertz) of the fifth equalizer band.
	writeGEQ5Freq(N)	Set the center frequency for the fifth equalizer band to N Hz. N is a 16 bit value.
	writeGEQ5Freq(500)	Set the center frequency for the fifth equalizer band to 500 Hz.
Band 5 Width	readGEQ5Width	Return the width (in tenths of octaves) of the fifth equalizer band.
	writeGEQ5Width(N)	Set the width of the fifth equalizer band to N/10 octaves. N is a 16 bit value.
	writeGEQ5Width(20)	Set the width of the fifth equalizer band to 2 octaves.
Band 5 Level	readGEQ5Level	Return the level setting (in tenths of dB) of the fifth equalizer band.
	writeGEQ5Level(N)	Set the level of the fifth equalizer band to N/10 dB. N is a 16 bit value.
	writeGEQ5Level(0)	Set the level of the fifth equalizer band to 0 dB.

Register	Alpha Code	Description
Band 6 Center Frequency	readGEQ6Freq	Return center frequency (in Hertz) of the sixth equalizer band.
	writeGEQ6Freq(N)	Set the center frequency for the sixth equalizer band to N Hz. N is a 16 bit value.
	writeGEQ6Freq(1000)	Set the center frequency for the sixth equalizer band to 1 kHz.
Band 6 Width	readGEQ6Width	Return the width (in tenths of octaves) of the sixth equalizer band.
	writeGEQ6Width(N)	Set the width of the sixth equalizer band to N/10 octaves. N is a 16 bit value.
	writeGEQ6Width(20)	Set the width of the sixth equalizer band to 2 octaves.
Band 6 Level	readGEQ6Level	Return the level setting (in tenths of dB) of the sixth equalizer band.
	writeGEQ6Level(N)	Set the level of the sixth equalizer band to N/10 dB. N is a 16 bit value.
	writeGEQ6Level(0)	Set the level of the sixth equalizer band to 0 dB.
Band 7 Center Frequency	readGEQ7Freq	Return center frequency (in Hertz) of the seventh equalizer band.
	writeGEQ7Freq(N)	Set the center frequency for the seventh equalizer band to N Hz. N is a 16 bit value.
	writeGEQ7Freq(2000)	Set the center frequency for the seventh equalizer band to 2 kHz.
Band 7 Width	readGEQ7Width	Return the width (in tenths of octaves) of the seventh equalizer band.
	writeGEQ7Width(N)	Set the width of the seventh equalizer band to N/10 octaves. N is a 16 bit value.
	writeGEQ7Width(20)	Set the width of the seventh equalizer band to 2 octaves.
Band 7 Level	readGEQ7Level	Return the level setting (in tenths of dB) of the seventh equalizer band.
	writeGEQ7Level(N)	Set the level of the seventh equalizer band to N/10 dB. N is a 16 bit value.
	writeGEQ7Level(0)	Set the level of the seventh equalizer band to 0 dB.
Band 8 Center Frequency	readGEQ8Freq	Return center frequency (in Hertz) of the eight equalizer band.
	writeGEQ8Freq(N)	Set the center frequency for the eight equalizer band to N Hz. N is a 16 bit value.
	writeGEQ8Freq(4000)	Set the center frequency for the eight equalizer band to 4 kHz.
Band 8 Width	readGEQ8Width	Return the width (in tenths of octaves) of the eight equalizer band.
	writeGEQ8Width(N)	Set the width of the eight equalizer band to N/10 octaves. N is a 16 bit value.
	writeGEQ8Width(20)	Set the width of the eight equalizer band to 2 octaves.
Band 8 Level	readGEQ8Level	Return the level setting (in tenths of dB) of the eight equalizer band.
	writeGEQ8Level(N)	Set the level of the eight equalizer band to N/10 dB. N is a 16 bit value.
	writeGEQ8Level(0)	Set the level of the eight equalizer band to 0 dB.
Band 9 Center Frequency	readGEQ9Freq	Return center frequency (in Hertz) of the ninth equalizer band.
	writeGEQ9Freq(N)	Set the center frequency for the ninth equalizer band to N Hz. N is a 16 bit value.
	writeGEQ9Freq(8000)	Set the center frequency for the ninth equalizer band to 8 kHz.

Register	Alpha Code	Description
Band 9 Width	readGEQ9Width	Return the width (in tenths of octaves) of the ninth equalizer band.
	writeGEQ9Width(N)	Set the width of the ninth equalizer band to N/10 octaves. N is a 16 bit value.
	writeGEQ9Width(20)	Set the width of the ninth equalizer band to 2 octaves.
Band 9 Level	readGEQ9Level	Return the level setting (in tenths of dB) of the ninth equalizer band.
	writeGEQ9Level(N)	Set the level of the ninth equalizer band to N/10 dB. N is a 16 bit value.
	writeGEQ9Level(0)	Set the level of the ninth equalizer band to 0 dB.
Band 10 Center Frequency	readGEQ10Freq	Return center frequency (in Hertz) of the tenth equalizer band.
	writeGEQ10Freq(N)	Set the center frequency for the tenth equalizer band to N Hz. N is a 16 bit value.
	writeGEQ10Freq(16000)	Set the center frequency for the tenth equalizer band to 16 kHz.
Band 10 Width	readGEQ10Width	Return the width (in tenths of octaves) of the tenth equalizer band.
	writeGEQ10Width(N)	Set the width of the tenth equalizer band to N/10 octaves. N is a 16 bit value.
	writeGEQ10Width(20)	Set the width of the tenth equalizer band to 2 octaves.
Band 10 Level	readGEQ10Level	Return the level setting (in tenths of dB) of the tenth equalizer band.
	writeGEQ10Level(N)	Set the level of the tenth equalizer band to N/10 dB. N is a 16 bit value.
	writeGEQ10Level(0)	Set the level of the tenth equalizer band to 0 dB.

TABLE G-3 **Graphic Equalizer Alpha Code (Channel-wise coefficients)**

Register	Alpha Code	Description
Mode	readGEQMode	Indicate if Graphic Equalizer is enabled or disabled.
	writeGEQModeDisable	Disable Graphic Equalizer.
	writeGEQModeEnable	Enable Graphic Equalizer.
Use	readGEQUse	Return the value of the Use Control Register.
	writeGEQUseDisable	Disable the Use Control Register
	writeGEQUseEnable	Enable the Use Control Register
Parameter Change	readGEQparamChg	Return the current value of the parameter change register.
	writeGEQparamChgRamp	Signal a change to parameters and require a ramped change to coefficients.
	writeGEQparamChgNoRamp	Signal a change to parameters without requiring a ramped change to coefficients.
Number of Bands	readGEQnumBands	Return number of active equalizer bands.
	writeGEQnumBands(N)	Set the number of active bands to N, where N is a decimal value, inclusive. Only values between 2 and the lower of 12 and the value of "Max Bands" as shown in Table G-6 are valid.
	writeGEQnumBands(10)	Set the number of active bands to 10.

Register	Alpha Code	Description
Center Frequency	readGEQCh<**p**>Ba<**q**>Freq	Return center frequency (in Hertz) of the q-th equalizer band for the p-th audio channel data. Audio channels are numbered 0 through 15 and bands are numbered 1 through 12. So, readGEQCh0Ba1Freq will return the center frequency for the first band of the left channel equalizer[a].
	writeGEQCh<**p**>Ba<**q**>Freq(N)	Set the center frequency for q-th equalizer band for the p-th audio channel data to N Hz. N is a 16 bit value.
	writeGEQCh<p>Ba1Freq(32) writeGEQCh<p>Ba2Freq(63) writeGEQCh<p>Ba3Freq(125) writeGEQCh<p>Ba4Freq(250) writeGEQCh<p>Ba5Freq(500) writeGEQCh<p>Ba6Freq(1000) writeGEQCh<p>Ba7Freq(2000) writeGEQCh<p>Ba8Freq(4000) writeGEQCh<p>Ba9Freq(8000) writeGEQCh<p>Ba10Freq(16000)	Set the center frequency for the first equalizer band to 32 Hz. Set the center frequency for the second equalizer band to 63 Hz. Set the center frequency for the third equalizer band to 125 Hz. Set the center frequency for the fourth equalizer band to 250 Hz. Set the center frequency for the fifth equalizer band to 500 Hz. Set the center frequency for the sixth equalizer band to 1 kHz. Set the center frequency for the seventh equalizer band to 2 kHz. Set the center frequency for the eight equalizer band to 4 kHz. Set the center frequency for the ninth equalizer band to 8 kHz. Set the center frequency for the tenth equalizer band to 16 kHz. Default center frequency for each numbered band is held constant across all channels. Default center frequency for the 11th and 12th bands are undefined since the number of active bands is set to 10 by default.
Width	readGEQCh<**p**>Ba<**q**>Width	Return the width (in tenths of octaves) of the q-th equalizer band for the p-th audio channel data. Audio channels are numbered 0 through 15 and bands are numbered 1 through 12. So, readGEQCh0Ba1Width will return the center frequency for the first band of the left channel equalizer.
	writeGEQCh<**p**>Ba<**q**>Width(N)	Set the width of the q-th equalizer band for the p-th audio channel data to N/10 octaves. N is a 16 bit value.
	writeGEQCh<p>Ba1Width(20) writeGEQCh<p>Ba2Width(20) writeGEQCh<p>Ba3Width(20) writeGEQCh<p>Ba4Width(20) writeGEQCh<p>Ba5Width(20) writeGEQCh<p>Ba6Width(20) writeGEQCh<p>Ba7Width(20) writeGEQCh<p>Ba8Width(20) writeGEQCh<p>Ba9Width(20) writeGEQCh<p>Ba10Width(20)	Set the width of all 10 active bands for all the channels to 2 octaves. Default width for the 11th and 12th bands are undefined since the number of active bands is set to 10 by default.
Level	readGEQCh<**p**>Ba<**q**>Level	Return the level setting (in tenths of dB) of the q-th equalizer band for the p-th audio channel data. Audio channels are numbered 0 through 15 and bands are numbered 1 through 12. So, readGEQCh0Ba1Level will return the setting for the first band of the left channel equalizer.
	writeGEQCh<**p**>Ba<**q**>Level(N)	Set the level of the q-th equalizer band for the p-th audio channel data to N/10 dB. N is a 16 bit value.
	writeGEQCh<p>Ba1Level(0) writeGEQCh<p>Ba2Level(0) writeGEQCh<p>Ba3Level(0) writeGEQCh<p>Ba4Level(0) writeGEQCh<p>Ba5Level(0) writeGEQCh<p>Ba6Level(0) writeGEQCh<p>Ba7Level(0) writeGEQCh<p>Ba8Level(0) writeGEQCh<p>Ba9Level(0) writeGEQCh<p>Ba10Level(0)	Set the level of all the bands for all the channels to 0 dB. Default level for the 11th and 12th bands are undefined since the number of active bands is set to 10 by default.

[a] For a mapping between channel numbers and logical audio channels refer to `paftyp_a.h`

G.1.1 Mode

The GEQ3 Mode Control Register controls basic operation of the GEQ3 Algorithm. If zero, operation is disabled, and if non-zero, operation is enabled.

G.1.2 Use Control

The GEQ3 Use Control Register provides the ability to enable and disable the GEQ3 Algorithm.

If the Use Control Register is set using writeGEQUseDisable then the GEQ3 Algorithm is not applied to the input.

G.1.3 Parameter Change

The GEQ3 Parameter Change Register is used to communicate to the GEQ3 Algorithm that one or more of its control registers have been changed externally. Changes to parameters will not have any effect on the output of GEQ3 until after a parameter change has been signalled and acknowledged. Parameter change can be signalled using writeGEQparamChgRamp or writeGEQparamChgNoRamp.

When writeGEQparamChgRamp is used, GEQ3 recalculates all coefficients and “ramps them in”; i.e., it applies the changed coefficients in small steps spread over the complete number of samples present in an audio frame. This approach avoids audio artifacts but will consume a larger number of CPU cycles for one frame of audio.

When writeGEQparamChgNoRamp is used, the newly calculated coefficients are applied all at once. This avoids problems that may arise on account of a sudden change in CPU load when writeGEQparamChgRamp is used, but care should be taken that parameters are not changed in big jumps to avoid the possibility of audio artifacts.

The GEQ3 Algorithm resets the value of the Parameter Change Register to zero to indicate that changes to parameters have been effected. Normally, the response to readGEQparamChg indicates 0; if it indicates otherwise, the latest command has not yet been effected.

G.1.4 Number of Bands

The maximum number of bands that can be filtered by GEQ3 is set at link time via the GEQ3 initialization parameter set (see Section G.1.9). At run-time, the number of bands can be reduced below that maximum number. The GEQ3 Number of Bands Register specifies the currently active number of equalizer bands. This value may only be set to a value between 2 and the maximum number; setting the Number of Bands Register to a value outside this range results in undefined operation. Note that any change to this register must be followed by writing a non-zero value to the Parameter Change register, for the change to be effective.

Changes to this register (followed by setting the Parameter Change Register) during audio playback may result in audio artifacts. The user is advised to mute the audio before flagging a parameter change to the GEQ3 Algorithm (via the Parameter Change Register) following a change to the value of this register, and unmute the audio only after the change has taken effect. Care must also be taken to ensure that the Center Frequency, Width and Level Register settings for all the bands that the user intends to activate have been correctly set.

Band 1 is always a low shelving filter, and the last Band is always a high shelving filter (where ‘last’ means the band having number equal to the Number of Bands register).

G.1.5 Center Frequency

A Center Frequency Register is associated with each of the equalizer bands. This register specifies the cut-off frequency (in Hertz) to be used for the low- and high-shelf filters associated with the first and last equalizer bands respectively and the center frequency (in Hertz) to be used for the presence filters associated with each of the other equalizer bands.

Changes to these registers (followed by setting the Parameter Change Register) during audio playback may result in audio artifacts. It is recommended to mute the audio before applying drastic changes to this category of registers.

G.1.6 Width

A Width Register if associated with each of the equalizer bands. This register specifies the width (in tenths of octaves) of the filter associated with a particular band. The GEQ3 Algorithm divides this value by 10 to obtain the value of the width in octaves: e.g., to set the width to 1.5 octaves, the corresponding register must be set to a value of 15.

Changes to these registers (followed by setting the Parameter Change Register) during audio playback may result in audio artifacts.

Since the first and the last bands utilize shelving filters, the width control doesn’t have any effect on these bands.

Note that in parametric EQ calculation of GEQ3 the band width in octaves is between midpoint (Level/2) gain frequencies and bandwidth is not between -3dB frequencies. For example, if we configure GEQ3 with Center frequency f_c= 900 Hz and Level = +10dB and Width = 2 Octaves [Q = 0.6666] we will observe a gain of +10dB at f_c= 900 Hz and gain of +5dB at f_l= 450 Hz and f_h= 1800 Hz

Design Considerations

f_c - Center frequency

f_l - Lower cut off frequency

f_h - Upper cut off frequency

BW - Bandwidth in Hertz

N - Bandwidth in octaves

Q - quality factor

$\boldsymbol{BW} = \boldsymbol{f}_{\mathrm{h}}\text{-}\boldsymbol{f}_{\mathrm{l}}$

$\boldsymbol{f}_{\mathrm{c}} = \mathrm{sqrt}(\boldsymbol{f}_{\mathrm{h}}*\boldsymbol{f}_{\mathrm{l}})$

$\boldsymbol{Q = f_{c}/BW = 2^{width/20}/(2\wedge^{width/10} - 1)}$

$\boldsymbol{f}_{\mathrm{h}} = \boldsymbol{f}_{\mathrm{c}}*2\wedge(\boldsymbol{N}/2);$

$\boldsymbol{f}_{\mathrm{l}} = \boldsymbol{f}_{\mathrm{c}}*\ 2\wedge(-\boldsymbol{N}/2);$

$\boldsymbol{N} = \log2(\boldsymbol{f}_{\mathrm{h}}/\boldsymbol{f}_{\mathrm{l}});$

Example: For a filter with center frequency $\boldsymbol{f}_{\mathrm{c}}$= 1000Hz and Quality factor of $\boldsymbol{Q}$=0.66, $\boldsymbol{N}$ can be calculated as

$\boldsymbol{BW}$ = 1500Hz,

by solving $\boldsymbol{f}_{\mathrm{h}}\ \text{-}\boldsymbol{f}_{\mathrm{l}}$ = 1500 and $\boldsymbol{f}_{\mathrm{l}}*\boldsymbol{f}_{\mathrm{h}}$ = (100)^2, $\boldsymbol{f}_{\mathrm{l}}$ =500Hz and $\boldsymbol{f}_{\mathrm{h}}$= 2000Hz

$\boldsymbol{N}$= log2(2000/500) = 2   (Note that GEQ3 accepts 10*$\boldsymbol{N}$ as width parameter).

TABLE G-4 **GEQ Relation between Quality Factor (Q) and N**

Quality Factor (Q)	N
0.5	2.543106606
1.0	1.388483827
1.5	0.944821416
2.0	0.714037274
2.5	0.573298474
3.0	0.478699344
3.5	0.410809269
4.0	0.359741049
4.5	0.319942868
5.0	0.288060261

G.1.7 Level

A Level Register is associated with each of the equalizer bands. This register specifies the boost or cut (in tenths of dB) to applied to that band. The GEQ3 Algorithm divides this value by 10 to obtain the value of the level in dB: e.g., to set the level to -3.3 dB, the corresponding register must be set to a value of -33.

These registers may be changed during audio playback (and the change communicated to the GEQ3 Algorithm by setting the Parameter Change Register) without risk of audio artifacts.

If the level of any band is increased above 0, clipping may occur. The developer should take care to provide enough headroom in the implementation to avoid clipping, by utilizing the Master Volume Control or Volume Trim Control Registers to decrease the level such that full range input will not exceed the full range output. For more information on Volume Control Usage see Chapter 5.4 (Volume Control).

The degree of precision in the filter transfer function decreases with filter width and filter center frequency. As a general rule, you may expect less than +/-0.1 dB deviation, unless using narrow-width filters at low fc. Table N-5 shows the deviation resulting from narrow filters at low fc, whether the filter is single- or double- precision.[55]

TABLE G-5 **Maximum filter deviation (dB) from expected plot**

		fc					
		20	**25**	**30**	**40**	**50**	**60**
width	**1**	0.6	0.4	0.06	0.06	0.11	
	2	0.06	0.06	0.16	0.06		
	3	0.06	0.2	0.15	0.07		
	4	0.06	0.17	0.06	0.06		
	5	0.06	0.13	0.06	0.09		
	6	0.06	0.1	0.06	0.1		
	7	0.06		0.11			
	8	0.06					
	9	0.1					
	10	0.1		**less than 0.1 dB**			
	11	0.1					
	12	0.1					
	13						

G.1.8 Loudness Compensation Filter

When instantiated with the parameter named `IGEQ_PARAMS_LOU`, GEQ3 behaves as a loudness compensation filter. The response characteristics of this filter are designed to closely imitate those of the human ear. Thus, enabling this algorithm has the effect of providing a more intuitive volume control.

G.1.8.1 Loudness Compensation Mode

The LOU Mode Control Register controls basic operation of the LOU Algorithm. If zero, operation is disabled, and if non-zero, operation is enabled.

[55] Deviations obtained with 10 bands, with Band 2 set to indicated width and fc. Remaining bands set to width = 20. All levels set to 0.

G.1.8.2 Loudness Compensation Bypass Control

The LOU Bypass Control Register provides the ability to enable and disable the LOU Algorithm.

If the Bypass Control Register is set using writeGEQLOUBypassEnable then the LOU Algorithm *is not* applied to the input.

If the Bypass Control Register is set using writeGEQLOUBypassDisable then the LOU Algorithm *is* applied to the input.

G.1.9 Parameter Sets

Configuration parameters allow for link-time control over the operation of Audio Stream Processing (ASP) algorithms. GEQ3 uses this mechanism to provide control over:

- the maximum number of bands supported
- the maximum number of channels supported
- channel-wise coefficients, and
- high-precision filtering

The SDK user chooses from among available parameter sets by specifying the parameter set of choice in the ASP link line in the *-patchs.c file.

G.1.9.1 Channel-wise Coefficients

By default, GEQ3 applies the same center frequency, level and width for each band across all channels for that band. This is referred to as “ganged coefficient” operation, since the same coefficients are used for all channels. Alternatively, GEQ3 can be instantiated with “channel-wise coefficients” so as to allow for independent control (of center frequency, level, and width) over each channel and band. Such control comes at the cost of additional memory.

If GEQ3 is instantiated with “channel-wise coefficients”, and its set of initialization parameters is changed so that the maximum number of bands is *not* 10, then the #definition of GEQ_MAXNUMBANDS in geq3_a.h must be changed from 10 to match the new value.

G.1.9.2 High-Precision Filters

By default, GEQ3 uses single precision filters for all bands -- filter coefficients, accumulator and states are stored as 32-bit IEEE floating point numbers. High-precision filtering uses 64-bit IEEE floating point numbers to store the accumulator and filter states. High-precision filtering may be selected for one or more bands. It comes at a higher cost of memory and processing power.

When using High-Precision Filters for some bands and not for other bands, use the High-Precision Filters for the lowest-frequency filters (i.e., set Band1 to the lowest frequency, Band2 to the next-lowest frequency, etc.). This will reduce the Total Harmonic Distortion (THD) at low frequencies. THD for mid and high frequencies is already very low, and using High-Precision filters on those bands is not warranted.

TABLE G-6 **Graphic Equalizer Parameter Sets**

Parameter	Channels	Bands	Ganged coefficients	High-precision Filters
IGEQ_PARAMS_EQX	8	10	Yes	None
IGEQ_PARAMS_EQX_HQ	8	10	Yes	First 4
IGEQ_PARAMS_EQX_16CHN	16	10	No	None
IGEQ_PARAMS_EQT	8	3	Yes	None
IGEQ_PARAMS_EQV	8	5	Yes	None
IGEQ_PARAMS_LOU	8	1	Yes	None
IGEQ_PARAMS_EQX_16CHN_UNICOEF	16	10	Yes	None
IGEQ_PARAMS_EQX_HQ_16CHN_UNICOEF	16	10	Yes	First 4

G.1.10 Usage of Audio Frame Data Structure Registers

A pointer to the Audio Frame Data Structure is passed to an ASP Algorithm as part of both the reset and apply function. It encapsulates a “frame” of audio sample data on which processing is to take place.

Table G-7 (Usage of Audio Frame Data Structure Quantities) illustrates the registers in the Audio Frame Data Structure which are utilized by the GEQ3 Algorithm and whether the registers are read, written, or read *and* written.

TABLE G-7 **Usage of Audio Frame Data Structure Quantities**

Audio Frame Data Structure Register	Read/Written
channelConfigurationStream	R
sampleRate	R
sampleCount	R
data.sample	R/W
sampleProcess	R/W

The GEQ3 Algorithm supports all sampling rates supported by PA/F. When operating as LOU, only the sample rates of 32kHz, 44.1kHz, 48kHz, 64kHz, 88.2kHz, and 96kHz are supported.

If the incoming Sample Rate register changes, then the coefficients for the GEQ3 processing are updated.

The GEQ3 field of the Sample Process register is set when GEQ3 processing has successfully occurred.

Note: If GEQ3 is configured for ganged coefficients, do not use the alpha commands for channel-wise coefficients. Similarly, if GEQ3 is configured for channel-wise coefficients, do not use the alpha commands for ganged coefficients.

If GEQ3 is configured for N bands, do not use alpha commands for band M > N.

APPENDIX H Bass Management Algorithm (BM2)

The Bass Management (BM2) Algorithm is an Audio Stream Processing (ASP) Algorithm with an Application Interface (API) that permits one to exercise control over various aspects of its operation. This appendix explains the details of this component of the Performance Audio Framework (PA/F) API. A block diagram of BM2 processing is provided in Figure H-1.

BM2 implements bass management as defined by Dolby (see the References below). Some of the terminology in this Appendix assumes familiarity with the references.

The Bass Management (BM2) Algorithm differs from the Bass Management (BM1) Algorithm in the following ways:

- BM1 has a single filter cut-off frequency for all filtered and redirected channels, whereas BM2 provides control for different cut-off frequencies for Main (front left, right), Center, Surrounds (surround left, right), Back (back left, right), LFE, and Subw channels.
- BM2 includes gain control for the bass redirected from satellite channels (on a channel-pair basis); BM1 does not.

BM2 uses the term “cutoff frequency” to mean the “corner frequency”. For second-order filters, the “cutoff frequency” is down by 3dB. For fourth-order filters, the “cutoff frequency” is down by 6dB.

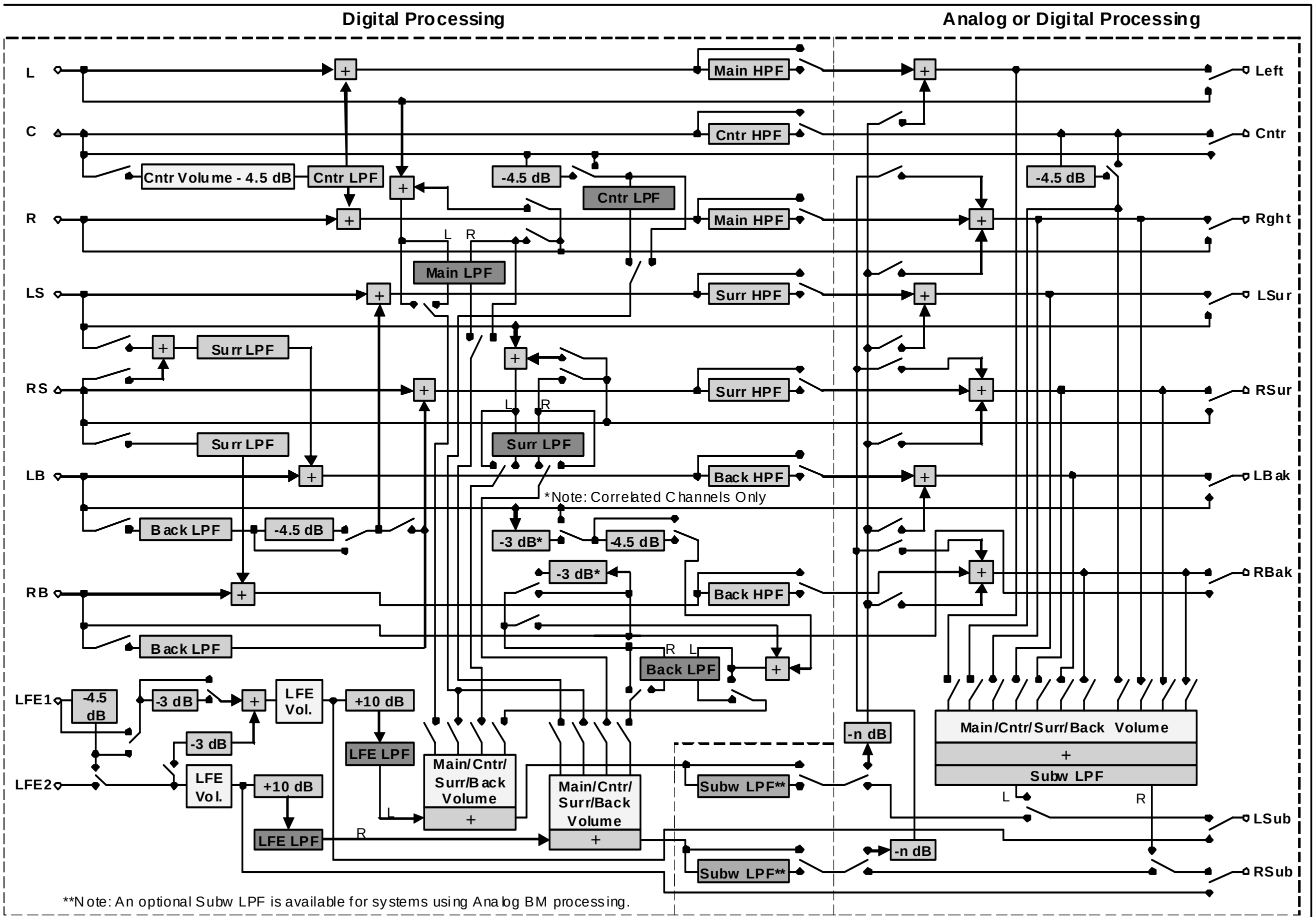
The System Stream (“SYS”) component of PA/F, when operating in the Auto Recreation Mode, configures some of the Bass Management Algorithm registers automatically as a function of the SYSSpeaker register settings. This capability is intended to provide suitable default bass redirection functionality, but may not satisfy all possible configurations. For details, see Section H.5 (SYS and BM).

BM2 supports the configuration of 0, 1, or 2 subwoofers. When two subwoofers are configured, BM2 assumes that they are located left/right; front/back subwoofer positioning is not supported.

H.0.1 References

1. Dolby Licensee Information Manual: Dolby Digital Consumer Decoder, Issue 5
2. Dolby Bass Management Implementation Guide, Issue 1;

FIGURE H-1 **BM2 Block Diagram**



*Note: Correlated Channels Only

**Note: An optional Subw LPF is available for systems using Analog BM processing.

In this block diagram:

- the LPF shown in green are *only* effective in Digital-only operation. In Analog operation, they are all-pass.
- the LPF shown in red are effective only in Analog operation, and only optionally effective even then.
- bass redirection attenuations shown in yellow are adjustable via Volume trim.
- processing of Wide and High channels is not shown, since it would increase complexity even further

H.1 Control, Status, and Command Registers

Table H-1 (Bass Management Alpha Code (Read/Write registers)) gives a list of alpha code symbols for the BM2 Algorithm. For an explanation of the various forms of register usage, see Section 7.1 (Register Architecture).

Please keep the following in mind when reading this table:

- Default settings for control registers are shaded. Default settings result at power-up or reset.
- Bit-mapped registers allow the specification of multiple values simultaneously using the logical-or of any combination of the command or response for that register.
- Alpha code of type `write` is used to set values in registers, and alpha code of type `read` is used to get values from registers.
- Numeric values for the preprocessor symbols shown in this table and appendix are provided in C-language form in the alpha code symbol file `P:/alpha/bm2_a.h`.

The BM2 Algorithm has no command registers.

BM2 has additional configuration options beyond those of BM1, but the extra flexibility can initially appear overwhelming. To reduce the complexity, the list of supported alpha commands has been separated into three sections:

- **read/write** registers **-** including the “Select” registers (Section H.2)
- **read-only** registers **-** including the “Status” registers (Section H.3)
- **convenient combinations** (Section H.4).

The discussion starts with the **read/write** registers, because these are the registers that must be set up correctly for BM2 operation. The **read-only** registers exist primarily for debug. **Convenient combinations** provide for a way to set and/or read multiple registers in one alpha command; in some cases their mnemonic names can make BM2 setup more natural.

H.2 Read/Write Registers

The following registers control BM2’s operation.

Register - Significance
Mode - select 2nd-order, 4th-order, or auto-order low-pass filtering
Bypass - temporarily disable BM2 operation, so that it can be reenabled subsequently
OutputConfigurationAuto - enable automatic recognition of ProLogic operation and back-channel split
OutputConfigurationSelectAuto - manual recognition of ProLogic operation and back-channel split
OutputConfigurationSelectOCNumber - Dolby Bass Management Output Configuration number
OutputConfigurationSelectChannelsLowFrequency - bitmap indicating full-range output channels
OutputConfigurationSelectAncillary - select digital/analog and local bass options

Register - Significance
Filter Cutoff Main - specify corner frequency for main speaker filtering
Filter Cutoff Center - specify corner frequency for center speaker filtering
Filter Cutoff Surround - specify corner frequency for surround speaker filtering
Filter Cutoff Back - specify corner frequency for back speaker(s) filtering
Filter Cutoff LFE - specify corner frequency for LFE input filtering
Filter Cutoff Subwoofer - specify corner frequency for subwoofer output filtering
Main Volume - volume trim for any low frequency content redirected from the main channels
Center Volume- volume trim for any low frequency content redirected from the center channel
Surround Volume - volume trim for any low frequency content redirected from the surround channels
Back Volume - volume trim for any low frequency content redirected from the back channel(s)
LFE Volume- volume trim for LFE input

TABLE H-1 **Bass Management Alpha Code (Read/Write registers)**

Register	Alpha Code	Description
Mode (0x04)	readBMMode	Indicate if Bass Management is enabled or not. If enabled, indicate the Low Pass Filter order selection.
	writeBMModeDisable	Disable Bass Management.[a]
	writeBMModeLPF2ndOrder	Enable Bass Management with second-order low pass filtering. See Section H.2.1.1 (Second-Order Low Pass Filter Mode vs. Fourth-Order Low Pass Filter Mode).
	writeBMModeLPF4thOrder	Enable Bass Management with fourth-order low pass filtering. See Section H.2.1.1 (Second-Order Low Pass Filter Mode vs. Fourth-Order Low Pass Filter Mode).
	writeBMModeLPFAuto	Enable Bass Management and automatically select the appropriate-order low pass filtering. See Section H.2.1.1 (Second-Order Low Pass Filter Mode vs. Fourth-Order Low Pass Filter Mode)
Bypass (0x12)	readBMBypass	Indicate whether Bass Management is bypassed or not. See Section H.2.1 (Mode and Bypass).
	writeBMBypassDisable	Perform Bass Management processing.
	writeBMBypassEnable	Bypass Bass Management processing.

Register	Alpha Code	Description
Output Configuration Select OC Number (0x0b)	readBMOCSelectOCNumber	Return BM Output Configuration Select Output Configuration Number.
	writeBMOCSelectOCNumberNone	No output configuration selected.
	writeBMOCSelectOCNumberDOC0	Dolby Output Configuration 0 selected.
	writeBMOCSelectOCNumberDOC1	Dolby Output Configuration 1 selected
	writeBMOCSelectOCNumberDOC2	Dolby Output Configuration 2 selected.
	writeBMOCSelectOCNumberDOC3	Dolby Output Configuration 3 selected.
	writeBMOCSelectOCNumberDOCAuto	Automatic output configuration.
Output Configuration Select Channels Low Frequency (0x08)	readBMOCSelectChannelsLowFreq	Return BM Output Configuration Select Channels Low Frequency Register value. This is a bit mapped 8-bit register.
	writeBMOCSelectChannelsLowFreqMain	Main channels are capable of bass reproduction.
	writeBMOCSelectChannelsLowFreqCntr	Center channel is capable of bass reproduction.
	writeBMOCSelectChannelsLowFreqSurr	Surround channels are capable of bass reproduction.
	writeBMOCSelectChannelsLowFreqBack	Back channel(s) is(are) capable of bass reproduction.
	writeBMOCSelectChannelsLowFreqWide	Wide channels are capable of bass reproduction.
	writeBMOCSelectChannelsLowFreqHigh	High channels are capable of bass reproduction.
	writeBMOCSelectChannelsLowFreqN(*NN*)	Set BM Output Configuration Select Channels Low Freq Register to 0x*NN*, where 0x*NN* is an 8-bit bit-mapped value. Setting a bit to ‘1’ indicates the associated channel(s) can reproduce bass, as associated with a “Large” speaker. bit 0: Reserved; set to ‘0’ bit 1: Main channels bit 2: Center channel bit 3: Surround channels bit 4: Back channel(s) bit 5: Wide channels bit 6: High[b] channels If (a) channel(s) is not present, its corresponding bit(s) is (are) ignored.

Register	Alpha Code	Description
Output Configuration Select Ancillary (0x09)	readBMOCSelectAncillary	Indicate what Supplemental Output Configuration features are enabled or not. This is a bit mapped 8-bit register. Setting a bit to ‘1’ indicates the associated supplemental aspect is enabled unless otherwise indicated. bit 0 : Subwoofer Low Pass Filter bit 1 : Subwoofer Reinforcement bit 2 : Local Bass Front Mode ‘0’ = Non-local, ‘1’ = Local bit 3 : Local Bass Rear Mode ‘0’ = Non-local, ‘1’ = Local bit 4 : Local Bass Master Enable bit 5-6: Reserved; set to ‘0’ bit 7 : Digital-Only BM implementation Description of these ancillary features is provided in Section H.2.2.3 (Output Configuration Select Ancillary Register).
	writeBMOCSelectAncillaryDgtlLBNoneStd (0x81)	Set ancillary output configuration to: - Digital-Only BM implementation - Local Bass Disabled - Subwoofer reinforcement disabled (Dolby standard configuration)
	writeBMOCSelectAncillaryDgtlLBNoneAlt (0x83)	Set ancillary output configuration to: - Digital-Only BM implementation - Local Bass Disabled - Subwoofer reinforcement enabled (Dolby “Alternate” configuration)
	writeBMOCSelectAncillaryDgtlLBFrntStd (0x95)	Set ancillary output configuration to: - Digital-Only BM implementation - Local Bass Enabled - Local Bass Front =Local, Rear = Non-local - Subwoofer reinforcement disabled
	writeBMOCSelectAncillaryDgtlLBFrntAlt (0x97)	Set ancillary output configuration to: - Digital-Only BM implementation - Local Bass Enabled - Local Bass Front = Local, Rear = Non-local - Subwoofer reinforcement enabled
	writeBMOCSelectAncillaryDgtlLBRearStd (0x99)	Set ancillary output configuration to: - Digital-Only BM implementation - Local Bass Enabled - Local Bass Front = Non-local, Rear = Local - Subwoofer reinforcement disabled
	writeBMOCSelectAncillaryDgtlLBRearAlt (0x9b)	Set ancillary output configuration to: - Digital-Only BM implementation - Local Bass Enabled - Local Bass Front = Non-local, Rear = Local - Subwoofer reinforcement enabled
	writeBMOCSelectAncillaryDgtlLBBothStd (0x9d)	Set ancillary output configuration to: - Digital-Only BM implementation - Local Bass Enabled - Local Bass Front = Local, Rear = Local - Subwoofer reinforcement disabled
	writeBMOCSelectAncillaryDgtlLBBothAlt (0x9f)	Set ancillary output configuration to: - Digital-Only BM implementation - Local Bass Front enabled - Local Bass Front = Local, Rear = Local - Subwoofer reinforcement enabled

Register	Alpha Code	Description
Output Configuration Select Ancillary *(continued)*	writeBMOCSelectAncillaryAnlgLBNone-SubwStd (0x00)	Set ancillary output configuration to: - Digital/Analog BM implementation - Local Bass Disabled - Subwoofer filter disabled (Dolby standard)
	writeBMOCSelectAncillaryAnlgLBNone-SubwFil (0x01)	Set ancillary output configuration to: - Digital/Analog BM implementation - Local Bass Disabled - Subwoofer filter enabled
	writeBMOCSelectAncillaryAnlgLBFrntSub-wStd (0x14)	Set ancillary output configuration to: - Digital/Analog BM implementation - Local Bass Enabled - Local Bass Front = Local, Rear = Non-local - Subwoofer filter disabled
	writeBMOCSelectAncillaryAnlgLBFrntSub-wFil (0x15)	Set ancillary output configuration to: - Digital/Analog BM implementation - Local Bass Enabled - Local Bass Front = Local, Rear = Non-local - Subwoofer filter enabled
	writeBMOCSelectAncillaryAnlgLBRear-SubwStd (0x08)	Set ancillary output configuration to: - Digital/Analog BM implementation - Local Bass Enabled - Local Bass Front = Non-local, Rear = Local - Subwoofer filter disabled
	writeBMOCSelectAncillaryAnlgLBRear-SubwFil (0x19)	Set ancillary output configuration to: - Digital/Analog BM implementation - Local Bass Enabled - Local Bass Front = Non-local, Rear = Local - Subwoofer filter enabled
	writeBMOCSelectAncillaryAnlgLBBoth-SubwStd (0x1c)	Set ancillary output configuration to: - Digital/Analog BM implementation - Local Bass Enabled - Local Bass Front = Local, Rear = Local - Subwoofer filter disabled
	writeBMOCSelectAncillaryAnlgLBBoth-SubwFil (0x1d)	Set ancillary output configuration to: - Digital/Analog BM implementation - Local Bass Enabled - Local Bass Front = Local, Rear = Local - Subwoofer filter enabled
	writeBMOCSelectAncillaryN(NN)	Specify the value of the SelectAncillary register directly.

Register	Alpha Code	Description
Output Configuration Auto (0x05)	readBMOCAuto	Indicates the current settings of this bit-mapped 8-bit register. The default setting for this register is indicted by one or more entries below to indicate the bit flags that are set to ‘1’.
	writeBMOCAutoDisable	Disables automatic recognition of ProLogic operation and Correlated back channels. See Section H.2.2.1 for information on this feature.
	writeBMOCAutoProLogic	Enable Automatic recognition of ProLogic operation.
	writeBMOCAutoCorrelation	Enable Automatic recognition of Correlated back channels
	writeBMOCAutoN(*NN*)	Set BM Output Configuration Auto Register to 0x*NN*, where 0x*NN* is an 8-bit bit-mapped value. Setting a bit to ‘1’ indicates that the associated automatic recognition is enabled. bit 0: Auto ProLogic recognition bit 1: Auto Correlation recognition bit 2-7: Reserved; set to ‘0’
	writeBMOCAutoN(3)	Enable Auto ProLogic and Correlation recognition.

Register	Alpha Code	Description
Output Configuration Select Auto (0x0a)	readBMOCSelectAuto	Indicates the current settings of this bit-mapped 8-bit register. The default setting for this register is indicted by one or more entries below to indicate the bit flags that are set to ‘1’.
	writeBMOCSelectAutoDisable	See Section H.2.2.5 (Output Configuration Select Auto Register) for information on this feature. If auto PL2X recognition is enabled, this setting has no effect. If auto PL2X recognition is disabled (via the OC Auto register), this setting allows BM2 to redirect bass from surrounds and back channels, even when those channels were generated by PL2X. --------- If auto Correlation recognition is enabled, this setting has no effect. If auto Correlation recognition is disabled (via the OC Auto register), this setting allows BM2 to redirect bass from correlated backs without attenuating for any correlation.
	writeBMOCSelectAutoProLogic	If Auto ProLogic recognition is enabled (via the OCAuto register), this setting has no effect. If Auto ProLogic recognition is disabled (via the OCAuto register), this setting prevents BM2 from redirecting surround/back bass.
	writeBMOCSelectAutoCorrelation	If Auto Correlation is enabled, this setting has no effect. If Auto Correlation recognition is disabled (via the OCAuto register), this setting prevents compensation for correlated back channel content.
	writeBMOCSelectAutoN(*NN*)	Set BM Output Configuration Select Auto Register to 0x*NN*, where 0x*NN* is an 8-bit bit-mapped value. Setting a bit to ‘1’ indicates the associated fallback facet is enabled. bit 0: Manual Pro Logic recognition bit 1: Manual Correlation recognition bit 2-7: Reserved; set to ‘0’

Register	Alpha Code	Description
Filter Cutoff Control (Satellite channels)	readBMFcMain (0x18,0x19)	Return BM filter cutoff frequency value in Hz for Main channels.
	writeBMFcMainN(*NN*)	Write BM Filter Cutoff value in Hz for Main channels. This value is quantized to a multiple of 10 Hz within the supported frequency range {40,50, ..., 200}. See example in Section H.7 (Examples).
	writeBMFcMainN(80)	Set BM Filter Cutoff to 80 Hz for Main channels.
	readBMFcCntr (0x1a,0x1b)	Return BM filter cutoff frequency value in Hz for Center channel.
	writeBMFcCntrN(*NN*)	Write BM Filter Cutoff value in Hz for Center channel. This value is quantized to a multiple of 10 Hz within the supported frequency range {40,50 .. 200}. See example in Section H.7 (Examples).
	writeBMFcCntrN(80)	Set BM Filter Cutoff to 80 Hz for Center channel.
	readBMFcSurr (0x1c,0x1d)	Return BM filter cutoff frequency value in Hz for Surround channels.
	writeBMFcSurrN(*NN*)	Write BM Filter Cutoff value in Hz for Surround channels. This value is quantized to a multiple of 10 Hz within the supported frequency range {40,50 .. 200}. See example in Section H.7 (Examples).
	writeBMFcSurrN(80)	Set BM Filter Cutoff to 80 Hz for Surround channels.
	readBMFcBack (0x1e,0x1f)	Return BM filter cutoff frequency value in Hz for Back channel(s).
	writeBMFcBackN(*NN*)	Write BM Filter Cutoff value in Hz for Back channel(s). This value is quantized to a multiple of 10 Hz within the supported frequency range {40,50 .. 200}. See example in Section H.7 (Examples).
	writeBMFcBackN(80)	Set BM Filter Cutoff to 80 Hz for Back channel(s).
	readBMFcWide (0x34,0x35)	Return BM filter cutoff frequency value in Hz for Wide channels.
	writeBMFcWideN(*NN*)	Write BM Filter Cutoff value in Hz for Wide channels. This value is quantized to a multiple of 10 Hz within the supported frequency range {40,50 .. 200}. See example in Section H.7 (Examples).
	writeBMFcWideN(80)	Set BM Filter Cutoff to 80 Hz for Wide channels.
	readBMFcHigh (0x36,0x37)	Return BM filter cutoff frequency value in Hz for High channels.
	writeBMFcHighN(*NN*)	Write BM Filter Cutoff value in Hz for High channel(s). This value is quantized to a multiple of 10 Hz within the supported frequency range {40,50 .. 200}. See example in Section H.7 (Examples).
	writeBMFcHighN(80)	Set BM Filter Cutoff to 80 Hz for High channels.

Register	Alpha Code	Description
Filter Cutoff Control (Subwoofer channels)	readBMFcLFE	Return LFE Filter Cutoff control register.
	writeBMFcLFEN(*NN*) (0x28,0x29)	Write BM LFE Filter Cutoff control register for LFE low pass filter. The value written to this register determines the cutoff frequency for LFE channel filtering. NN = 0 for automatic setting.
	writeBMFcLFEN(0)	Set BM LFE Filter Cutoff control register for automatic setting of cutoff frequency. With this setting, the cutoff frequency of the LFE channel is set to the maximum cutoff frequency of “Small” satellite channels (Main, Cntr, Surr, Back). If all satellite channels are “Large,” a default value of 80 Hz is used.
	readBMFcSubw	Return Subwoofer Filter Cutoff control register.
	writeBMFcSubwN(*NN*) (0x2a,0x2b)	Write BM Subwoofer Filter Cutoff control register for Subwoofer low pass filter. The value written to this register determines the cutoff frequency for Subwoofer channel filtering. NN = 0 for automatic setting.
	writeBMFcSubwN(0)	Set BM Subwoofer Filter Cutoff control register for automatic setting of cutoff frequency. With this setting, the cutoff frequency of the Subwoofer channel is set to the maximum cutoff frequency of “Small” satellite channels (Main, Cntr, Surr, Back) and LFE channel. If all satellite channels are “Large,” the Subwoofer cutoff frequency is set equal to the LFE cutoff frequency.
Main Volume (Bass redirection trim) (0x30)	readBMMainVolume	Return current volume setting used for any redirection of Main (Left and Right) channels’ bass.
	writeBMMainVolumeN(0)	BM Main volume level is 0 dB.
	writeBMMainVolumeN(NN)	Set Main Volume to *NN*, where *NN* is an 8-bit value in units of 0.5 dB. For example, the alpha code writeBMMainVolumeN(-20) sets the Main Volume Control Register to -10 dB.
Center Volume (Bass redirection trim) (0x31)	readBMCntrVolume	Return current volume setting used for any redirection of Center channel bass.
	writeBMCntrVolumeN(0)	BM Center volume level is 0 dB.
	writeBMCntrVolumeN(NN)	Set Center Volume to *NN*, where *NN* is an 8-bit value in units of 0.5 dB. For example, the alpha code writeBMCntrVolumeN(-20) sets the Center Volume Control Register to -10 dB.
Surround Volume (Bass redirection trim) (0x32)	readBMSurrVolume	Return current volume setting used for redirection of Surround channel(s) bass.
	writeBMSurrVolumeN(0)	BM Surround volume level is 0 dB.
	writeBMSurrVolumeN(NN)	Set Surround Volume to *NN*, where *NN* is an 8-bit value in units of 0.5 dB. For example, the alpha code writeBMSurrVolumeN(-20) sets the Surround Volume Control Register to -10 dB.

Register	Alpha Code	Description
Back Volume (Bass redirection trim) (0x33)	readBMBackVolume	Return current volume setting used for redirection of Back channel(s) bass.
	writeBMBackVolumeN(0)	BM Back volume level is 0 dB.
	writeBMBackVolumeN(NN)	Set Back Volume to *NN*, where *NN* is an 8-bit value in units of 0.5 dB. For example, the alpha code writeBMBackVolumeN(-20) sets the Back Volume Control Register to -10 dB.
Wide Volume (Bass redirection trim) (0x3C)	readBMWideVolume	Return current volume setting used for redirection of bass from Wide channels.
	writeBMWideVolumeN(0)	BM Wide volume level is 0 dB.
	writeBMWideVolumeN(NN)	Set Wide Volume to *NN*, where *NN* is an 8-bit value in units of 0.5 dB. For example, the alpha code writeBMWideVolumeN(-20) sets the Wide Volume Control Register to -10 dB.
High Volume (Bass redirection trim) (0x3D)	readBMHighVolume	Return current volume setting used for redirection of bass from High channels.
	writeBMHighVolumeN(0)	BM High volume level is 0 dB.
	writeBMHighVolumeN(NN)	Set High Volume to *NN*, where *NN* is an 8-bit value in units of 0.5 dB. For example, the alpha code writeBMHighVolumeN(-20) sets the High Volume Control Register to -10 dB.
LFE Volume (Bass trim) (0x34)	readBMLFEVolume	Return current volume trim setting applied to LFE input.
	writeBMLFEVolumeN(0)	BM LFE volume level is 0 dB.
	writeBMLFEVolumeN(NN)	Set LFE Volume to *NN*, where *NN* is an 8-bit value in units of 0.5 dB. For example, the alpha code writeBMLFEVolumeN(-20) sets the LFE Volume Control Register to -10 dB.
Override Bass Collection Mask (0x10)	readBMOverrideBassCollectionMask	Return the current setting of the Override Bass Collection Mask bitfield.
	writeBMOverrideBassCollectionMaskNone	Set “no override”
	writeBMOverrideBassCollectionMaskMain	See Chapter H.2.6 (Overriding Bass Collection)
	writeBMOverrideBassCollectionMaskCntr	
	writeBMOverrideBassCollectionMask-MainCntr	
	writeBMOverrideBassCollectionMaskN(N)	

[a] Once disabled, bass management should not be re-enabled. Attempting to do so will result in undefined operation of BM processing. For effective bypass of BM processing, see Section H.2.1 (Mode and Bypass).

[b] “High” channels are also known as “Head” channels. Specifically, these refer to Lh and Rh, but not Ch (also known as Cvh).

H.2.1 Mode and Bypass

The BM Mode Register controls basic operation of the BM2 Algorithm. If zero, BM operation is disabled; if non-zero, operation is enabled. Note that disabling BM2 via `writeBMModeDisable` is supported, but subsequent re-enabling of BM2 via

- `writeBMModeEnable`, or
- `writeBMModeLPF2ndOrder`, or
- `writeBMModeLPF4thOrder`, or
- `writeBMModeLPFAuto`

is not supported.

If the Bass Management Algorithm is disabled (via `writeBMModeDisable`), none of

- bass redirection,
- filtering, nor
- LFE volume control

is performed. The subwoofer output channel is a direct copy of the LFE input.

BM2 may be enabled in one of three modes: *Low Pass Filter Second-Order, Low Pass Filter Fourth-Order, or Low Pass Filter Auto*. These three modes are discussed in more detail in section P.2.1 below. The command `writeBMModeEnable` has a numerical value equivalent to the command `writeBMModeLPF2ndOrder`, but BM is enabled via any of the `writeBMModeLPFxxxx` commands.

The BM Bypass Register provides a means to temporarily disable BM, so that it can subsequently be reenabled. The BM Mode Control Register should ***not*** be used for this purpose; once disabled via `writeBMModeDisable`, Bass Management must not be re-enabled, and attempting to do so will result in undefined operation of BM processing.

If the Bass Management Algorithm is enabled (via `writeBMModeEnable` or `writeBMModeLPFxxx`, together with `writeBMBypassDisable`) but no output configuration number is selected (e.g., `writeBMSelectOCNumberNone`), no filtering nor redirection of the bass signals to or from the satellite channels is performed. However, the BM LFE Volume Control Register *does* affect the subwoofer output generated from the LFE input. This effectively bypasses Bass Management processing, except that Bass Management performs adjustment of the LFE input to its proper level. This state can be selected by the alpha code sequence `writeSYSRecreationModeDirect` or `writeSYSRecreationModeDont`, then `writeBMOCSelectOCNumberNone`; see section Section H.5 for a discussion of SYS interaction with BM.

If the Bass Management Algorithm is enabled and an output configuration number is selected, filtering is enabled and bass signals are redirected from the input channels to the output channels according to the output configuration and speaker size settings. This redirection includes level control of the LFE input as indicated by the BM LFE Volume Control Register.

H.2.1.1 Second-Order Low Pass Filter Mode vs. Fourth-Order Low Pass Filter Mode

Setting the BM Mode Control Register using `writeBMModeLPF2ndOrder` selects *Second-Order Low Pass Filter* mode. This mode employs second-order high pass filtering and second-order low pass filtering.

Setting the BM Mode Control Register using `writeBMModeLPF4thOrder` selects *Fourth-Order Low Pass Filter* mode. This mode employs second-order high pass filtering and fourth-order low pass filtering.

Dolby recommends that second-order low pass filtering be used for speaker configurations LSxx0 and SLxx0.

Setting the BM Mode Control Register using `writeBMModeLPFAuto` automatically selects 2nd-order low pass filtering for speaker configurations LSxx0 and SLxx0, and 4th-order low pass filtering for all other speaker configurations. See the description of the ModeStatus register in Table H-8 (Bass Management Alpha Code (Read-Only registers))

Switching between *Second-Order Low Pass Filter, Fourth-Order Low Pass Filter,* and *Low Pass Filter Auto* modes is allowed.

H.2.2 Output Configuration

Five 8-bit registers are used to select the Output Configuration:

- Select Output Configuration Number (Section H.2.2.1 (Output Configuration Select OC Number Register))
- Select Channels Low Frequency (Section H.2.2.2 (Output Configuration Select Channels Low Frequency Register))
- Select Ancillary (Section H.2.2.3 (Output Configuration Select Ancillary Register))
- Auto Processing (Section H.2.2.4 (Output Configuration Auto Processing))
- Select Auto Processing (Section H.2.2.5 (Output Configuration Select Auto Register))

All of these registers except the fourth (“Auto Processing”) are setup by SYS if it is operating - see Section H.5 (SYS and BM). If SYS is ***not*** operating, the first three and the last one of these registers can most easily be set via use of the Section H.4 (Convenient combinations) writeBMOCSelect_XmXcXsXb_Ysub alpha command.

H.2.2.1 Output Configuration Select OC Number Register

Dolby identifies an output configuration numbering scheme in References 1 and 2. BM2 adheres to the Dolby numbering scheme, which is summarized as:

OC Number	resulting BM processing	typical speaker configuration
0	Sum all satellites into LFE for SW output. No satellite filtering occurs.	none
1	Sum satellites into LFE for SW output. High-pass filter satellites, low-pass filter SW.	SSSS1
2	Sum satellites (other than L/R) with LFE, redirect sum to L/R.	LSSSx
3	Redirect C bass to L/R. Redirect LFE to satellites other than C.	LSLLx

TABLE H-2

TABLE H-3 **Output Configuration Select OC Number Control Register**

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Invalid Config.	None	Reserved		Dolby Output Configuration Number			
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

This register is used to select the Dolby Output Configuration number. BM2 uses this configuration number to determine how to move low-frequency content from channels which can not reproduce it to channels that can reproduce it.

When SYS is enabled to perform BM configuration, SYS continually overwrites this register with an appropriate setting, based on the SYSSpeaker settings; see Section H.5.

Bits 3 through 0 are a four-bit field for selecting Dolby Output Configuration 0 through 3 and, alternatively, DOCAuto. ‘No configuration’ is selected by setting Bit 6. Bit 7 is an internal system indicator that a configuration error exists for *automatic* selection of output configuration.

Host selection of output configuration should be restricted to setting desired values for bits 0 through 3 and 6. The remainder of bits should be set to ‘0’. For example, selection of Dolby Output Configuration 2 is achieved by setting this register to 0x02. Selection of no output configuration is achieved by setting this register to 0x40. Alpha codes for setting this register are provided in Table H-1 (Bass Management Alpha Code (Read/Write registers)).

When Doc0 is selected, BM does Doc0 processing and almost nothing else: all satellite channels are passed through, and a scaled copy of each satellite is added to LFE to form the subwoofer output. The only optional processing is that of subwoofer filtering, which can be selected via the OCSelectAncillary Subwoofer LPF bit.

When Doc3 is set, BM effectively assumes that the Main channels are low-frequency capable, even if the Main bit is not set in the OCSelectChannelsLowFrequency register.

When Doc1 or Doc2 is set, BM processing is effectively driven by the settings of the OCSelectChannelsLowFreq and OCSelectAncillary registers.

When DOCAuto is set, BM determines which Dolby Output Configuration is appropriate, based on the requested channel configuration and the OCSelectChannelsLowFreq register.

H.2.2.2 Output Configuration Select Channels Low Frequency Register

The BM Output Configuration Select Channels Low Frequency Control Register is a bit-mapped register which determines channel filtering and redirection of bass content. The bit indicators within this register are shown in Table P-4 (Output Configuration Select Channels Low Frequency Register). A channel (pair)’s bit should be set to 1 if the associated speaker(s) are capable of reproducing bass (i.e., are “Large”).

TABLE H-4 **Output Configuration Select Channels Low Frequency Register**

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Reserved	High	Wide	Back	Surround	Center	Main	Reserved
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

When SYS is enabled to perform BM configuration, SYS continually overwrites this register with an appropriate setting, based on the SYSSpeaker settings. See Section H.5.

Host selection of bass reproducing channels should be restricted to setting desired values for bits associated with available channels. Bits associated with unavailable channels are ignored. Reserved bits should be set to ‘0’. For example, selection of Main and Surround channels as bass reproducing channels is achieved by setting this register to 0x0a.

If Doc3 is selected, the Main bit in this register is effectively set.

H.2.2.3 Output Configuration Select Ancillary Register

The BM Output Configuration Select Ancillary Control Register is a bit-mapped register. which enables/disables some BM features. The bit positions within this register are shown in Table H-5 (Output Configuration Select Ancillary Control Register). Setting an ancillary control bit to ‘1’ enables the corresponding feature.

TABLE H-5 **Output Configuration Select Ancillary Control Register**

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Digital Only	Reserved		Local Bass Enable	Local Bass Rear	Local Bass Front	Subw. Reinforce	Subw. LPF
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	0	0	0	0	0

Bits and their significance are indicated in the following list:

- *Digital Only Bit* - selects an all-digital implementation of BM processing, as described in sections 2.2 and 3.2 of Reference 2. The Digital Only bit should be set if analog hardware is *not* being used for any aspects of BM processing. Set this bit if BM2 should perform the functions identified in Figure P-1 as “Analog or Digital Processing”. With this bit cleared to 0, BM2 performs only the functions identified as “Digital Processing” in Figure H-1 (BM2 Block Diagram), and external analog circuitry is assumed present.[56]
- *Subwoofer Low Pass Filter (LPF) Bit* - selects filtering of the subwoofer in the digital domain for digital/analog Bass Management implementations. This control has no effect while the Digital Only bit is selected. The standard setting of Subwoofer Low

Pass Filter for digital/analog implementations is specified by Dolby as disabled. This is indicated by the Dolby specification in its various digital/analog output configuration illustrations. Note that this capability exists also with Dolby Output Configuration 0 (DOC0).

- *Subwoofer Reinforcement Bit* - selects bass reproduction using the subwoofer in addition to bass reproduction on other channels. This control has effect only when the Digital Only bit is selected; else it is effectively cleared. Subwoofer reinforcement is sometimes referred to as "double bass." The method implemented is illustrated by Dolby Output Configuration 2 with optional subwoofer present: LFE and bass redirected from Small channels is combined and distributed to Large channels, and then bass from Large channels is copied to the Subwoofer..
- *Local Bass Enable Bit* - is a master enable control for engaging Local Bass Front Mode and Local Bass Rear Mode which are defined below. The latter are mode selections that are only in effect while Local Bass Enable is asserted. The default setting of Local Bass Enable is deasserted. When asserted, Local Bass Enable enables both mode selections.
  - *Local Bass Front Mode bit* - determines the destination of center bass when the center channel is not low frequency capable (i.e., center is set to "small"). The default setting is "non-local", in which case bass from the center channel is redirected as per Dolby's guidance in Ref. 2 (i.e., Doc3[57] redirects center bass to L/R, Doc2 redirects center bass to all large speakers, Doc1 redirects center bass to Sw). Enabling "local bass front mode" results in center channel bass information being redirected to the left and right channels even when Doc1 or Doc2 is selected (so long as L/R are "Large"); thus the 'local' setting "keeps the bass local to the front."

    Special Case: if Dolby Output Configuration 3 ("Doc3") is selected, and Local Bass Front is set to "non-local," and Local Bass Enable is asserted, then the "Doc3 standard" redirection of center bass to left and right channels is ***not*** performed. Instead, center bass is redirected to subwoofer (if available) or (if not) to large channel(s). This is a special case, because normally setting Doc3 is sufficient to cause C bass to be directed to Left/Right.
  - *Local Bass Rear Mode Bit* - determines the destination of surround and back channel bass if one or more of the "rear" channels is not low frequency capable. This feature is complex in implementation, however it can be described as being similar to Local Bass Front Mode in that it "keeps the bass local to the rear." The default setting is disabled, which selects the subwoofer as the destination for the bass information of surround or back channels that are not low frequency capable. If the subwoofer is not present, the destination is becomes the available large channel(s). Setting Local Bass Rear Mode to "local" selects surround/back channels that are low frequency capable as the destination for bass from back/surround

[56] This terminology can be confusing, because when one "sets" the "Digital Only" bit, BM performs both the "Digital Processing" and "Analog or Digital Processing" shown in Figure H-1 (BM2 Block Diagram). Perhaps "All Digital" would be a better name, but "Digital Only" is used for historical reasons.

[57] "Doc3" is shorthand for "Dolby Output Configuration 3". See Section H.2.2.1 (Output Configuration Select OC Number Register)

channels that are not low frequency capable. Two examples are described next to help in understanding this feature:

a. Consider a speaker configuration having a single “small” speaker for the back channel and two “large” speakers for the surround channels. With Local Bass Enable asserted and Local Bass Rear Mode set to “local,” BM will redirect the bass from the back channel to the surround channels. Level adjustment is automatically provided as the back channel is “split” from one channel to two channels for ensuring equivalent output level.

b. Consider a speaker configuration having two “small” speakers for the back channel and two “large” speakers for the surround channels. With Local Bass Enable asserted and Local Bass Rear Mode set to “local,” BM will redirect the bass from the left back channel to the left surround channel, and redirect the bass from the right back channel to the right surround channel.

Reserved bits should be set to ‘0’. Alpha codes for setting this register are provided in Table H-1 (Bass Management Alpha Code (Read/Write registers)).

H.2.2.4 Output Configuration Auto Processing

BM2 can automatically adjust its operation to help the system meet bass redirection requirements established by Dolby and DTS.

It is recommended that BM2 be operated with writeBMOCAutoN(*3*) at all times, enabling automatic ProLogic and Correlation adjustment.

TABLE H-6 **Bass Management Output Configuration Auto Control Register**

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Reserved						Corre-lation	Pro Logic
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

- **ProLogic Surround/Back Bass Redirection**

Dolby specifies that, when ProLogic (e.g., PL2X operating in 2-channel to many-channel mode) produces the surround/back channel content, “Bass should not be redirected to or from any of the Surround or Back channels in any configuration” [other than Game mode].

To enable automatic BM2 operation to meet this requirement, set the “ProLogic” bit in the OCAuto register.

- **Correlated Back Bass Redirection Attenuation**

For those situations wherein a decoder or ASP produces 6.x channels, but the channel request is 7.x, the decoder or ASP may split the center surround channel into two back channels at a reduced amplitude. When this happens, if BM2 needs to sum the backs’ bass in order to redirect, the sum must be attenuated if the output level is to be correct.

To enable automatic adjustment in the event of back bass redirection, set the “Correlation” bit in the OCAuto register.

H.2.2.5 Output Configuration Select Auto Register

The BM Output Configuration Select Auto provides manual control for Surround/Back bass redirection and back bass attenuation. The settings are effective only when the corresponding OCAuto setting is disabled (thus disabling automatic operation). This register is used primarily for testing purposes, since use of automatic operation is highly recommended.

TABLE H-7 **Output Configuration Select Auto Register**

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Reserved						Correlation	Pro Logic
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

BM2’s behavior regarding bass redirection to/from surround/back channels is as follows:

Source of Surround/Back channels	OCAuto PL bit	OCSelectAuto PL bit	effect
Not PL2X	ON	don’t care	redirect
PL2X in Game mode	ON	don’t care	redirect
PL2X in other mode	ON	don’t care	don’t redirect
don’t care	OFF	ON	don’t redirect
don’t care	OFF	OFF	redirect

In the above table, “redirect” means that bass is redirected to/from the surround/back speakers if otherwise appropriate (i.e., if consistent with speaker size settings).

BM2 attenuates bass redirected from back channels as follows:

Back channel content	OCAuto Corr bit	OCSelectAuto Corr bit	effect
independent	ON	don’t care	no attenuation
equal	ON	don’t care	attenuate
don’t care	OFF	ON	attenuate
don’t care	OFF	OFF	no attenuation

Back channels are typically ‘equal’ if a single back channel is generated and then split for Lb/Rb. Decoders and ASPs which generate Lb/Rb in this manner provide an indication in the audio frame.

Note that the OCSelectAuto register is continually overwritten with ‘0’ so long as SYSRecreationModeAuto processing continues; see Section H.5. In this case, if the OCAuto PL bit is ***not*** set, surround/back bass redirection *will* occur, even if this violates Dolby’s dictum. Similarly, if the OCAuto Correlation bit is *not* set, bass redirected from backs is ***not*** attenuated, even if that would be appropriate.

H.2.3 Filter Cutoff

> The low-pass filters shown in green in Figure H-1 (BM2 Block Diagram) operate ***only*** when BM is set to Digital-Only. When BM is set to Digital/Analog, then these LPF blocks do not low-pass filter the inputs.

The BM Filter Cutoff Control Registers control the cutoff frequencies at which the BM filters operate. Available cutoff frequencies are multiples of 10 Hz for the range of 40 through 200 Hz. The value written to a BM Filter Cutoff Control Register is quantized to a multiple of 10 Hz and limited to the available cutoff frequencies. The corresponding BM Filter Cutoff Status Register (see Section H.3 (Read-only registers)) indicates the actual cutoff frequency used.

For example, the alpha code `writeBMFcMainN(108)` sets the BM Filter Cutoff Main Control Register to 108 Hz. The alpha code `readBMFcStatusMain` indicates the cutoff frequency selected for Main channels has been quantized (rounded down) and that BM filters for Main channels are operating with a 0x64 (100) Hz cutoff frequency. As presented in Section H.7 (Examples), values written to the BM Filter Cutoff Control Registers may be either hexadecimal or decimal.

A Filter Cutoff Control Register is provided for each satellite channel group: Main, Center, Surround and Back. All four of these registers may be written using one command (see Section H.4 (Convenient combinations)) or written separately using one command per register. In similar fashion, all four of these registers may be read using one command or read separately using one command per register.

A Filter Cutoff Control Register is provided for each subwoofer channel group: LFE and Subwoofer. These two registers may be written using one command (see Section H.4 (Convenient combinations)) or written separately using one command per register. In similar fashion, these two registers may be read using one command or read separately using one command per register.

LFE and Subwoofer Filter Cutoff:

The LFE filter cutoff is used to filter the LFE channel as part of the “bass summing” process. The Subwoofer filter cutoff is used to filter the Subwoofer signal before it is sent to the Subwoofer output. See Figure H-1 (BM2 Block Diagram) for location of these low pass filters in signal paths.

The values written to LFE Filter Cutoff and Subwoofer Filter Cutoff control registers determine the cutoff frequency for each channel:

0	Select automatic setting of cutoff frequency (default).
> 0	Manually set cutoff frequency, where the value is quantized to a multiple of 10 Hz within the supported frequency range {40, 50,.., 200}.

If automatic setting of LFE cutoff frequency is enabled: The cutoff frequency of the LFE channel is set to the maximum cutoff frequency of the "Small" satellite channels (Main, Cntr, Surr, Back). If all satellite channels are "Large," a default value of 80 Hz is used.

If automatic setting of Subwoofer cutoff frequency is enabled: The cutoff frequency of the Subwoofer channel is set to the maximum cutoff frequency of the "Small" satellite channels (Main, Cntr, Surr, Back) and the LFE channel. If all satellite channels are "Large," the Subwoofer cutoff frequency is set to the LFE cutoff frequency.

H.2.4 Bass Volume

The following control registers are provided for setting the level(s) used while summing channel(s) for redirecting bass information among the satellite and subwoofer output channels:

- BM Main Volume Control Register
- BM Center Volume Control Register
- BM Surround Volume Control Register
- BM Back Volume Control Register
- BM Wide Volume Control Register
- BM High Volume Control Register
- BM LFE Volume Control Register

The BM LFE Volume Control Register is a 16-bit (1-word) register. The representation used for these registers are the PA/F-standard volume register format, that is, a value in units of 0.5 dB. For example, the alpha code `writeBMLFEVolumeN(0)` sets the BM LFE Volume Control Register to 0 dB, and the alpha code `writeBMLFEVolumeN(-20)` sets the LFE Volume Control Register to -10 dB. The default value for all of these registers is 0 dB.

Caution: setting these registers to positive numerical values can result in clipping/saturation of the channels to which the bass information is redirected. BM2 does not currently comprehend these settings in the volume tracking scheme.

Certification Requirement:

Dolby requires that the value of the BM LFE Volume Control Register be within the range [-10 dB, 0 dB]. This requirement is not enforced in any way by the Performance Audio Framework, but rather it is assumed that this requirement will be met by any host which manipulates the contents of this register.

H.2.5 Volume Trim

In most situations when bass is redirected from a channel, a volume trim gain can be applied to that redirection. These trims are identified in yellow on Figure H-1 on page H-246; see ‘MainVolume’, ‘Cntr Volume’, ‘Surr Volume’, ‘Back Volume’ in Table H-1 on page H-248. Trim of the LFE input is provided via ‘LFE Volume’.

Note that when bass from the Center is redirected to the Main channels, the Cntr Volume trim is in addition to -4.5dB of built-in attenuation “for the split”.

H.2.6 Overriding Bass Collection

The information in this section is provided only for OEMs needing non-standard bass management operation. It can be skipped if the default of “writeBMOverrideBassCollectionMaskNone” is used.

In the BM2 operation sequence, after any local bass redirection[58], bass content is collected from "Small" channels[59] so that the summed bass content can be redirected to Subwoofer(s) and/or "Large" channel(s); the "Small" channels are then high-pass filtered[60].

The criterion as to whether to collect bass content from a channel (pair) can be inverted by setting the channel (pair) bit in the "BMOverrideBassCollectionMask" register.

To either:

[58] local bass redirection as enabled via BMOCSelectAncillary:LocalBassFront, BMOCSelectAncillaryLocalBassRear, or DOC3 operation

[59] "Large" channels are channels for which the appropriate bit is set in the BMOCSelectChannelsLowFrequency register. "Small" channels are channels for which the appropriate bit is *not* set in the BMOCSelectChannelsLowFrequency register.

[60] If DOC0 operation is selected, bass is not redirected, so overriding bass collection is not available.

- collect bass content from "Large" channel(s), or
- *not* collect bass content from "Small" channel(s),

set the appropriate channel (pair) bit in the BMOverrideBassCollectionMask.

Setting the Mask bit for "Large" channel(s) results in bass content being collected from the channel(s). Since "Large" channels are not high-pass-filtered, "extra" bass will typically result.

Setting the Mask bit for "Small" channel(s) results in bass content *not* being collected from the channel(s). Since "Small" channels are subsequently high-pass-filtered, bass content from those channels will be "lost".

In general, the default of writeBMOverrideBassCollectionMaskNone should be used, because setting a Mask bit typically results in bass level either higher or lower than appropriate, based on sound-power-level considerations.

As an example of nonzero mask usage, consider speaker configuration LLS1. LLS1 normally results in bass content from Ls and Rs being redirected to the subwoofer, with Ls and Rs being high-pass-filtered after bass redirection. By sending BMOverrideBassCollectionMaskFrntCntr, bass content from L, R, and C (in addition to Ls and Rs) will be redirected to the subwoofer, but only Ls and Rs will be high-pass-filtered.

Note that setting mask bits for "Large" channels (which results in "extra bass") does not have the same effect as that resulting from use of BMOCSelectAncillary:SubwooferReinforcement. With subwoofer reinforcement, bass content is first redirected from "Small" channels to "Large" channels, and then bass from large channels is additionally directed to the subwoofer. This can result in redirection level differences.

Since there are a large number of permutations as to speaker configuration, Mask bit settings, and program channel content, Texas Instruments highly recommends that any OEM using nonzero mask perform extensive testing of the resulting behavior to ensure that it behaves as desired. For example, comb effects will likely result if bass is collected from a channel and added-back into the same channel, due to filtering in the redirection path.

When using writeBMOverrideBassCollectionMaskN(N), use mask bits as identified in Table P-4 (Output Configuration Select Channels Low Frequency Register)

Successful use has been made of the following configurations:

Speaker Configuration	**Mask Bits**	**effect**
LLL1	LR	bass from L and R appears in the Subw output
LLS1	LR, C	bass from L, R, and C, in addition to Ls and Rs, appears in the Subw output
LSS1	LR	bass from L and R, in addition to C, Ls, and Rs, appears in the Subw output
LSL1	LR	bass from L and R, in addition to C, appears in the Subw output

H.3 Read-only registers

The BM2 registers identified in this table can be useful for diagnostic purposes.

Note that these registers are updated only while Bass Management is enabled and active (e.g., audio decoding is in progress and the ASP chain is enabled).

TABLE H-8 **Bass Management Alpha Code (Read-Only registers)**

Register	Alpha Code	Description
Status	readBMStatus	Return entire BM status structure.
ModeStatus (0x15)	readBMModeStatus	Valid only when writeBMModeLPFAuto is selected and BM2 is operating. Indicates whether 2nd-order or 4th-order Low-Pass filtering is being performed.
	wroteBMModeStatusLPF2ndOrder	
	wroteBMModeStatusLPF4thOrder	
	wroteBMModeStatusDisabled	BM2 has not operated in ModeLPFAuto since restart
Output Configuration Status Auto (0x0e)	readBMOCStatusAuto	Indicates the current settings of this bit-mapped 8-bit register. The default setting for this register is indicted by one or more entries below to indicate the bit flags that are set to ‘1’.
	wroteBMOCStatusAutoDisable	Effective operation is disabled for both the 1. ProLogic Surround/Back Bass Redirection disabler 2. Correlated Back Bass Redirection attenuation See Section H.2.2.1 for information on these two features.
	wroteBMOCStatusAutoProLogic	Redirection of ProLogic-generated Surround/Back Bass is disabled.
	wroteBMOCStatusAutoCorrelation	Correlated Back Bass redirection is attenuated if appropriate.
Output Configuration Status OC Number (0x0f)	readBMOCStatusOCNumber	Return BM Output Configuration Status OC Number Register value. The returned value indicates the output configuration number (independent of Pro Logic activity) that is engaged. This is a read-only register. See “Output Configuration Select OC Number Register” for more information.
	wroteBMOCStatusOCNumberNone	No output configuration is engaged.
	wroteBMOCStatusOCNumberDOC0	Dolby Output Configuration 0 is engaged.
	wroteBMOCStatusOCNumberDOC1	Dolby Output Configuration 1 is engaged.
	wroteBMOCStatusOCNumberDOC2	Dolby Output Configuration 2 is engaged.
	wroteBMOCStatusOCNumberDOC3	Dolby Output Configuration 3 is engaged.

Register	Alpha Code	Description
Output Configuration Status Channels Low Frequency (0x0c)	readBMOCStatusChannelsLowFreq	Indicate if a given channel pair is enabled for low frequency information (speaker is considered large) or not. This is a bit mapped 8-bit register.
	wroteBMOCStatusChannelsLowFreqMain	Main channels are engaged for bass reproduction.
	wroteBMOCStatusChannelsLowFreqCntr	Center channel is engaged for bass reproduction.
	wroteBMOCStatusChannelsLowFreqSurr	Surround channels are engaged for bass reproduction.
	wroteBMOCStatusChannelsLowFreqBack	Back channel(s) is engaged for bass reproduction.
Output Configuration Status Ancillary (0x0d)	readBMOCStatusAncillary	Indicate the supplemental aspects engaged. This is a bit mapped 8-bit register.
	wroteBMOCStatusAncillaryDgtlLBNoneStd	See Output Configuration Select Ancillary Register for detailed descriptions corresponding to these ancillary configurations.
	wroteBMOCStatusAncillaryDgtlLBNoneAlt	
	wroteBMOCStatusAncillaryDgtlLBFrntStd	
	wroteBMOCStatusAncillaryDgtlLBFrntAlt	
	wroteBMOCStatusAncillaryDgtlLBRearStd	
	wroteBMOCStatusAncillaryDgtlLBRearAlt	
	wroteBMOCStatusAncillaryDgtlLBBothStd	
	wroteBMOCStatusAncillaryDgtlLBBothAlt	
	wroteBMOCStatusAncillaryAnlgLBNoneSubwStd	
	wroteBMOCStatusAncillaryAnlgLBNoneSubwFil	
	wroteBMOCStatusAncillaryAnlgLBFrntSubwStd	
	wroteBMOCStatusAncillaryAnlgLBFrntSubwFil	
	wroteBMOCStatusAncillaryAnlgLBRearSubwStd	
	wroteBMOCStatusAncillaryAnlgLBRearSubwFil	
	wroteBMOCStatusAncillaryAnlgLBBothSubwStd	
	wroteBMOCStatusAncillaryAnlgLBBothSubwFil	
Filter Cutoff Status (Satellite channels)	readBMFcStatusMain (0x20,0x21)	Return BM filter cutoff frequency value in Hz for Main channels.
	readBMFcStatusCntr (0x22,0x23)	Return BM filter cutoff frequency value in Hz for Center channel.
	readBMFcStatusSurr (0x24,0x25)	Return BM filter cutoff frequency value in Hz for Surround channels.
	readBMFcStatusBack (0x26,0x27)	Return BM filter cutoff frequency value in Hz for Back channel(s).

Register	Alpha Code	Description
Filter Cutoff Status (Subwoofer channels)	readBMFcStatusLFE (0x2c,0x2d)	Return BM filter cutoff frequency value in Hz for LFE channel.
	readBMFcStatusSubw (0x2e,0x2f)	Return BM filter cutoff frequency value in Hz for Subwoofer channel.
Filter Rate Status (0x11)	readBMFilterRateStatus	Return the sample rate at which the BM filter is operating.This value is an index, corresponding with the write commands below. As filter coefficients are selected automatically, relative to sample rate, this register is only of interest should the user want to verify filter rate. This register is only updated while Bass Management is enabled and active, i.e. audio decoding is in progress.
	wroteBMFilterRateStatus32000Hz	32 kHz sample rate filter coefficients are engaged.
	wroteBMFilterRateStatus44100Hz	44.1 kHz sample rate filter coefficients are engaged.
	wroteBMFilterRateStatus48000Hz	48 kHz sample rate filter coefficients are engaged.
	wroteBMFilterRateStatus64000Hz	64 kHz sample rate filter coefficients are engaged.
	wroteBMFilterRateStatus88200Hz	88.2 kHz sample rate filter coefficients are engaged.
	wroteBMFilterRateStatus96000Hz	96 kHz sample rate filter coefficients are engaged.
	wroteBMFilterRateStatus128000Hz	128 kHz sample rate filter coefficients are engaged.
	wroteBMFilterRateStatus176400Hz	176.4 kHz sample rate filter coefficients are engaged.
	wroteBMFilterRateStatus192000Hz	192 kHz sample rate filter coefficients are engaged.
InactionReason (0x06)	readBMInactionReason	return an indication as to why BM2 is not operating
	wroteBMInactionReasonNone	no reason - should operate
	wroteBMInactionReasonModeDisabled	BMMode is Disabled
	wroteBMInactionReasonBypassEnabled	BMBypass is Enabled
	wroteBMInactionReasonActiveFailed	transition to Active failed (internal error)
	wroteBMInactionReasonSetupFailed	setup failed (internal error)
	wroteBMInactionReasonStreamUnknown	audio frame channel configuration is ‘unknown’
	wroteBMInactionReasonStreamNone	audio frame channel configuration is ‘none’
	wroteBMInactionReasonNoSubw	no subwoofer channel is allocated
	wroteBMInactionReasonSubwCount	audio frame subwoofer count is three or more
	wroteBMInactionReasonOCNONone	DOC0 is selected
	wroteBMInactionReasonBLKSIZE	audio frame’s sampleCount is larger than BLKSIZE for which BM2 was built
	wroteBMInactionReasonResetFailed	reset failed (internal error)

H.3.1 Output Configuration Status Auto Register

This read-only register indicates the “result” of the combination of the OCAuto register, OCSelectAuto register, and SYSRecreationMode in the operation of ProLogic Surround/ Back Bass Redirection and the Correlated Back Bass Redirection.

When the ProLogic bit is on, bass is not redirected from Surround/Back channels even if otherwise appropriate. Table H-9 (Definition of How the ProLogic Bit is Obtained) indicates how the ProLogic bit is obtained. In the table, ‘x’ indicates “don’t care”.

The audio frame’s SURRBASS indication is set by ProLogic when ProLogic expands Stereo to a channel configuration involving one or two Surround channels.

TABLE H-9 **Definition of How the ProLogic Bit is Obtained**

Bit in OCAuto	SYSRecreation-Mode	Bit in OCSelectAuto	Audio frame indication	Bit in OCStatusAuto	Effect
1	x	x	SURRBASS	1	no surround/back bass redirection
1	x	x	no SURRBASS	0	Surround/Back Bass is redirected if otherwise appropriate
0	an Auto mode	overwritten 0	x	0	Surround/Back Bass is redirected if otherwise appropriate
0	Direct or Don’t	1	x	1	no surround/back bass redirection
0	Direct or Don’t	0	x	0	Surround/Back Bass is redirected if otherwise appropriate

When the Correlation bit is on, any bass redirected from the two Back channels is attenuated. Table H-10 (Definition of How the Correlation Bit is Obtained) indicates how the Correlation bit is obtained. In the table, ‘x’ indicates “don’t care”.

The audio frame’s MONOBACK indication is set by a decoder or ASP that expands Cs to Lb/Rb via simple splitting/attenuating.

TABLE H-10 **Definition of How the Correlation Bit is Obtained**

Bit in OCAuto	SYSRecreation-Mode	Bit in OCSelectAuto	Audio frame indication	Bit in OCStatusAuto	Effect
1	x	x	MONOBACK	1	attenuate redirected back bass
1	x	x	no MONOBACK	0	redirected back bass is not attenuated
0	an Auto mode	overwritten 0	x	0	redirected back bass is not attenuated

Bit in OCAuto	SYSRecreation-Mode	Bit in OCSelectAuto	Audio frame indication	Bit in OCStatusAuto	Effect
0	Direct or Don’t	1	x	1	attenuate redirected back bass
0	Direct or Don’t	0	x	0	redirected back bass is not attenuated

H.3.2 Filter Rate Status register

The BM Filter Rate Status Register indicates the sample rate at which the BM filters are operating. As filter coefficients are selected automatically relative to audio sample rate, this register is only of interest should the user want to verify filter rate.

If an encoded bitstream indicates that the output audio has a sample rate Fs, BM uses Fs to select filtering coefficients.

H.3.3 Inaction Reason register

On occasion, BM2 will refuse to perform bass redirection due to some operational criterion not being met. In this situation, the InactionReason may provide useful information as to why this situation has arisen.

H.3.3.1 BLKSIZE

By default, the BM2 library that is provided in the SDK is built to support a maximum audio-frame sample count of 256. If an audio frame has sample count greater than 256, BM2 refuses to operate on it, and indicates this via ‘wroteBMInactionReasonBLKSIZE’.

Should an OEM need to operate with audio frame sample count larger than 256, BM2 must be instantiated with a params indication of the maximum sampleCount required, so that BM2 can request sufficient memory be allocated at initialization. Refer to the ‘params.c’ file in the final build. If an audio frame subsequently appears having a sampleCount larger than that for which BM2 was instantiated, BM2 will refuse to operate on it, and indicate this via ‘wroteBMInactionReasonBLKSIZE’.

H.4 Convenient combinations

The following alpha commands are useful for reading or writing multiple registers via single alpha command.

In some cases, the alpha command described here has a name which is more-descriptive of the desired operation than the individual alpha command identified in the other table sections. For example, sending writeBMOCSelect_SLSL1 sets all of the Output Configuration Select registers simultaneously to appropriate default settings for a speaker configuration of SLSL1.

TABLE H-11 **Bass Management Alpha Code (Convenient combinations)**

Register	Alpha Code	Description
Status	readBMStatus	Return entire BM status structure.
Output Configuration Select (0x08,0x09,0x0a,0x0b)	readBMOCSelect	Read entire BM Output Configuration Select Register value. This includes: BM Output Configuration Select OC Number, BM Output Configuration Select Auto, BM Output Configuration Select Ancillary and BM Output Configuration Select Channels Low Frequency Registers. Format of returned value: (Ancillary,ChannelsLowFreq),(OCNumber,Auto) Example: 0x8100,0x0102 Ancillary = 0x81, Channels=0x00, OC Number = 0x01, Auto = 0x02
	writeBMOCSelect_XmXcXsXbYsub	Manually engage a Dolby “Speaker setup” as described in Section H.4.1 (Selection of Digital-only Dolby “Speaker Setup” Configurations). These selections are default Digital-Only with Auto Correlation enabled. If a non-default Output Configuration Select Ancillary setting is desired, one should set it after having sent this alpha command. If a non-default Output Configuration Select Auto Setting is desired, one should set it after having sent this alpha command.

Register	Alpha Code	Description
Output Configuration Select OC Number and Output Configuration Select Auto Registers *(Combined) (0z0a,0x0b)*	readBMOCSelectOCAuto	Read value from both BM Output Configuration Select OC Number and BM Output Configuration Select Auto Registers. *If automatic selection of output configuration for a given speaker setup is active*, these registers are automatically set or may be inactive. They should not be written by the Host. *If inactive* and Output Configuration Auto Register (not Output Configuration Select Auto Register) is disabled, the write commands below allow direct selection of output configuration. For output configurations with Auto Correlation selected, Auto Correlation automatically takes effect only when appropriate. It is recommended that this feature always be selected as provided by the alpha codes below. Correlated channels are channels that have common signal content such that electrical summing of the signals requires level adjustment to maintain the same power level.
	writeBMOCSelectOCAutoNone	No output configuration selected.
	writeBMOCSelectOCAutoDOC0CorrelOff	Dolby Output Configuration 0 with Auto Correlation disabled.
	writeBMOCSelectOCAutoDOC0CorrelOn	Dolby Output Configuration 0 with Auto Correlation is selected.
	writeBMOCSelectOCAutoDPC0CorrelOff	Dolby Pro Logic Output Configuration 0 with Auto Correlation disabled.
	writeBMOCSelectOCAutoDPC0CorrelOn	Dolby Pro Logic Output Configuration 0 with Auto Correlation is selected.
	writeBMOCSelectOCAutoDOC1CorrelOff	Dolby Output Configuration 1 with Auto Correlation disabled.
	writeBMOCSelectOCAutoDOC1CorrelOn	Dolby Output Configuration 1 with Auto Correlation is selected.
	writeBMOCSelectOCAutoDPC1CorrelOff	Dolby Pro Logic Output Configuration 1 with Auto Correlation disabled.
	writeBMOCSelectOCAutoDPC1CorrelOn	Dolby Pro Logic Output Configuration 1 with Auto Correlation is selected.
	writeBMOCSelectOCAutoDOC2CorrelOff	Dolby Output Configuration 2 with Auto Correlation disabled.
	writeBMOCSelectOCAutoDOC2CorrelOn	Dolby Output Configuration 2 with Auto Correlation is selected.
	writeBMOCSelectOCAutoDPC2CorrelOff	Dolby Pro Logic Output Configuration 2 with Auto Correlation disabled.
	writeBMOCSelectOCAutoDPC2CorrelOn	Dolby Pro Logic Output Configuration 2 with Auto Correlation is selected.
	writeBMOCSelectOCAutoDOC3CorrelOff	Dolby Output Configuration 3 with Auto Correlation disabled.
	writeBMOCSelectOCAutoDOC3CorrelOn	Dolby Output Configuration 3 with Auto Correlation is selected.

Register	Alpha Code	Description
Filter Cutoff Control (Satellite channels)	readBMFcSatAll	Return BM filter cutoff frequency values in Hz for all satellite channels: Main, Center, Surround and Back.
	writeBMFcSatAllN(NN_1,NN_2,NN_3,NN_4)	Write independent BM Filter Cutoff value in Hz for each of all satellite channels: Main, Center, Surround and Back. This value is quantized to a multiple of 10 Hz within the supported frequency range {40,50 .. 200}. NN_1 = Main filter cutoff frequency NN_2 = Center filter cutoff frequency NN_3 = Surround filter cutoff frequency NN_4 = Back filter cutoff frequency See example in Section H.7 (Examples).
Filter Cutoff Control (Subwoofer channels)	readBMFcSubAll	Return values of LFE Filter Cutoff and Subwoofer Filter Cutoff control registers.
	writeBMFcSubAllN(NN_1,NN_2)	Write BM Filter Cutoff control registers for LFE and Subwoofer channels. The *NN* values written determine the cutoff frequency for each channel: 0 Select automatic setting of cutoff frequency. > 0 Manually set cutoff frequency, where the value is quantized to a multiple of 10 Hz within the supported frequency range {40, 50 .. 200}. The arguments are defined as follows: NN_1 = LFE Filter Cutoff control register NN_2 = Subwoofer Filter Cutoff control register See Figure H-1 (BM2 Block Diagram) for location of these low pass filters in signal paths. If automatic setting of LFE cutoff frequency is enabled: The cutoff frequency of LFE channel is set to the maximum cutoff frequency of “Small” satellite channels (Main, Cntr, Surr, Back). If all satellite channels are “Large,” a default value of 80 Hz is used. If automatic setting of Subwoofer cutoff frequency is enabled: The cutoff frequency of Subwoofer channel is set to the maximum cutoff frequency of “Small” satellite channels (Main, Cntr, Surr, Back) and LFE channel. If all satellite channels are “Large,” Subwoofer cutoff frequency is set equal to LFE cutoff frequency. See example in Section H.7 (Examples).

Register	Alpha Code	Description
Output Configuration Status OC Number and Output Configuration Status Auto Registers *(Combined) (0x0a,0x0b)*	readBMOCStatusOCAuto	Indicate the output configuration number engaged. For output configurations with Auto Correlation selected, Auto Correlation automatically takes effect only when appropriate. It is recommended that this feature always be selected. This register is only updated while Bass Management is enabled and active, i.e. audio decoding is in progress.
	wroteBMOCStatusOCAutoNoneCorrelOffPLOff	No output configuration is engaged. AutoCorrelation and Auto Pro Logic have no effect.
	wroteBMOCStatusOCAutoNoneCorrelOffPLOn	No output configuration is engaged. AutoCorrelation and Auto Pro Logic have no effect.
	wroteBMOCStatusOCAutoNoneCorrelOnPLOff	No output configuration is engaged. AutoCorrelation and Auto Pro Logic have no effect.
	wroteBMOCStatusOCAutoNoneCorrelOnPLOn	No output configuration is engaged. AutoCorrelation and Auto Pro Logic have no effect.
	wroteBMOCStatusOCAutoDOC0CorrelOff	Dolby Output Configuration 0 with Auto Correlation disabled.
	wroteBMOCStatusOCAutoDOC0CorrelOn	Dolby Output Configuration 0 with Auto Correlation enabled.
	wroteBMOCStatusOCAutoDPC0CorrelOff	Dolby Pro Logic Output Configuration 0 with Auto Correlation disabled.
	wroteBMOCStatusOCAutoDPC0CorrelOn	Dolby Pro Logic Output Configuration 0 with Auto Correlation enabled.
	wroteBMOCStatusOCAutoDOC1CorrelOff	Dolby Output Configuration 1 with Auto Correlation disabled.
	wroteBMOCStatusOCAutoDOC1CorrelOn	Dolby Output Configuration 1 with Auto Correlation enabled.
	wroteBMOCStatusOCAutoDPC1CorrelOff	Dolby Pro Logic Output Configuration 1 with Auto Correlation disabled.
	wroteBMOCStatusOCAutoDPC1CorrelOn	Dolby Pro Logic Output Configuration 1 with Auto Correlation enabled.
	wroteBMOCStatusOCAutoDOC2CorrelOff	Dolby Output Configuration 2 with Auto Correlation disabled.
	wroteBMOCStatusOCAutoDOC2CorrelOn	Dolby Output Configuration 2 with Auto Correlation enabled.
	wroteBMOCStatusOCAutoDPC2CorrelOff	Dolby Pro Logic Output Configuration 2 with Auto Correlation disabled.
	wroteBMOCStatusOCAutoDPC2CorrelOn	Dolby Pro Logic Output Configuration 2 with Auto Correlation enabled.
	wroteBMOCStatusOCAutoDOC3CorrelOff	Dolby Output Configuration 3 with Auto Correlation disabled.
	wroteBMOCStatusOCAutoDOC3CorrelOn	Dolby Output Configuration 3 with Auto Correlation enabled.

Register	Alpha Code	Description
Output Configuration Status	readBMOCStatus	Return updated output configuration value. The value is the same format as described above for readBMOCSelect. This register is only updated while Bass Management is enabled and active, i.e. audio decoding is in progress.
Filter Cutoff Status (Satellite channels)	readBMFcStatusSatAll	Return BM filter cutoff frequency value in Hz for each of all satellite channels: Main, Center, Surround and Back. These status registers are only updated while Bass Management is enabled and active, i.e. audio decoding is in progress.
Filter Cutoff Status (Subwoofer channels)	readBMFcStatusSubAll	Return BM filter cutoff frequency values in Hz for LFE and Subwoofer channels. See description of writeBMFcSubAllN(NN_1,NN_2) for associated information. These status registers are only updated while Bass Management is enabled and active, i.e. audio decoding is in progress.

H.4.1 Selection of Digital-only Dolby “Speaker Setup” Configurations

In cases wherein SYS is enabled to set bass management output configuration[61], use of the alpha commands described in this section is precluded, since SYS will continually overwrite.

Alpha codes are provided for selecting Dolby “Speaker setup” configurations as described in Reference 2. These alpha codes set the BM Output Configuration Select Register to appropriate values[62] as a function of speaker sizes. Similar to the “Speaker setups” described in Reference 2, appropriate configurations are explained below using this convention:

$X_mX_cX_sX_bY_w$ or $X_mX_cX_sX_bX_wX_hY_w$

where low frequency capability of channels is defined as:

X_m refers to Main channels: Small/Large (S/L)
X_c refers to Center channel: Small/Large/None (S/L/0)
X_s refers to Surround channels: Small/Large/None (S/L/0)
X_b refers to Back channels: Small/Large/None (S/L/0)
X_w refers to Wide channels: Small/Large/None (S/L/0)
X_h refers to Height channels: Small/Large/None (S/L/0)
Y_0 refers to Subwoofer channel: None/Bass1/Bass2 (0/1/2)

[61] See Section H.5 (SYS and BM)

[62] As recommended in Reference 2, Dolby speaker setups SLS0 and SL00 are not supported. Selecting these setups results in the "None" configuration (Bass Management is effectively bypassed).

For channels that are not available as outputs, as indicated by the channel configuration request, X_n is ignored.

By using an alpha command of the form writeBMOCSelect_XXXXXXY, one sets the value of the following registers:

- Output Configuration Select OC Number (see the table in Section H.6 (Dolby Specifications and Requirements))
- Output Configuration Select Channels Low Frequency (“L” channels’ bits are set)
- Output Configuration Select Ancillary (set to default value)
- Output Configuration Select Auto (set to default value)

Since `writeBMOCSelect_xxxxxxy` resets the contents of the SelectAncillary and SelectAuto registers to their default settings, one will have to modify those register settings after sending this alpha command if the default values are inappropriate.

H.5 SYS and BM

While the SYS stream is enabled for BM output configuration assistance, and SYSRecreationModeAuto is selected, the SYS component automatically controls three of the BM Output Configuration Select registers:

- OCSelectAuto
- OCSelectOCNumber
- OCSelectChannelsLowFrequency

SYS does this as a function of the SYSSpeakerMain, SYSSpeakerCntr, SYSSpeakerSurr, SYSSpeakerBack, and SYSSpeakerSubw settings. If your implementation does ***not*** use SYS for this function, you must set these three indicated Output Configuration Select registers as appropriate for your speaker configuration.

Note that when SYS is enabled to set BM’s output configuration, the three indicated OutputConfigurationSelect registers can be read via alpha command, but should not be written (since SYS will overwrite).

H.6 Dolby Specifications and Requirements

As mentioned in the introduction to this appendix, BM2 implements bass management as defined by Dolby. Basic functionality results from “speaker setup” as defined in Reference 2 Chapter 3 (that is, the bass redirection and filtering scheme is selected as a function of the speakers availability and their ability to reproduce low frequencies).

speaker config	associated Dolby Output Configuration (DOC)
LLL1	no redirection nor filtering
LLL0	DOC3, except that Center redirection and HPF are disabled
LLS1	DOC1, except that Main and Center redirection and HPF are disabled

speaker config	associated Dolby Output Configuration (DOC)
LLS0	DOC2, except that Center redirection and HPF are disabled
LL01	no redirection nor filtering
LL00	DOC2, except that Center redirection and HPF are disabled
LSL1	DOC3 with Subwoofer[a]
LSL0	DOC3
LSS1	DOC1, except that Main redirection and HPF are disabled
LSS0	DOC2
LS01	DOC1, except that Main redirection and HPF are disabled
LS00	DOC2
L0L1	no redirection nor filtering
L0L0	DOC3
L0S1	DOC1, except that Main redirection and HPF are disabled
L0S0	DOC2
L001	no redirection nor filtering
L000	DOC2
SLL1	DOC1, except that Center/Surround/Back redirection and HPF are disabled
SLL0	DOC2, except that Center redirection and HPF are disabled, and bass is redirected to Surround/Back instead of Main
SLS1	DOC1, except that Center redirection and HPF are disabled
SLS0	unsupported
SL01	DOC1, except that Center redirection and HPF are disabled
SL00	unsupported
SSL1	DOC1, except that Surround/Back redirection and HPF are disabled
SSL0	DOC2, except that bass is redirected to Surround/Back instead of Main
SSS1	DOC1
SSS0	DOC1, except that all redirection is disabled
SS01	DOC1
SS00	DOC1, except that all redirection is disabled
S0L1	DOC1, except that Surround/Back redirection and HPF are disabled
S0L0	DOC2, except that bass is redirected to Surround/Back instead of Main
S0S0	DOC1
S0S0	DOC1, except that all redirection is disabled
S001	DOC1
S000	DOC1, except that all redirection is disabled

[a] For LSL1, if you want Center bass be redirected to the Subwoofer instead of to the Mains, see See "Special Case" on Page 261

A limited number of options which modify basic functionality are available, primarily via the BMOCSelectAncillary register.

Some Dolby documents identify requirements for bass management which differ from those in Reference 2. Specifically, the Dolby Digital Multichannel Decoder Test Procedure, Issue 3 differs in the following regards:

- **A.** Section 4.4.3 Output Configuration 3 (LSL0) - Figure 4.5 - indicates that Output Configuration 3 should redirect LFE only to L and R at +5.5 dBr. *Instead, in this configuration, BM2 redirects LFE to L, R, Ls, and Rs, at -1 dBr.*
- **B.** Section 4.4.5 Other Conditions (LLL0) - row 4 - indicates that LFE should be redirected to all five satellite speakers at -1 dBr. *Instead, in this configuration, BM2 redirects LFE to L, R, Ls, and Rs only, at -1 dBr.*
- **C.** Section 4.4.5 Other Conditions (LLS0) - row 5 - indicates that LFE should be redirected to L, C, and R at +2.8dBr. *Instead, in this configuration, BM2 redirects LFE to L and R only, at +5.5 dBr.*

When queried regarding the disagreement between the documents, Dolby stated that:

> *... both the documents ( ) are "correct" in the sense that Dolby will accept several different approaches to these Bass Management configurations.*
>
> *Generally speaking, as long as the LFE signal is attenuated appropriately for the number of bass reproducers used in the system, the output will be acceptable.*

H.7 Examples

1. After boot, read the default values selected for BM Filter Cutoff Control Registers:

   Using `readBMFcSatAll`, 0x50 (80) Hz is indicated for each satellite channel group: Main, Center, Surround and Back channels:
   ```
   alpha 0xce40,0x1808,0x0050,0x0050,0x0050,0x0050
   ```

   Using `readBMFcMain`, 0x50 (80) Hz is indicated for Main channels:
   ```
   alpha 0xcb40,0x0018,0x0050
   ```

   Using `readBMFcCntr`, 0x50 (80) Hz is indicated for Center channel:
   ```
   alpha 0xcb40,0x001a,0x0050
   ```

   Using `readBMFcSurr`, 0x50 (80) Hz is indicated for Surround channels:
   ```
   alpha 0xcb40,0x001c,0x0050
   ```

   Using `readBMFcStatusBack`, 0x50 (80) Hz is indicated for Back channel(s):
   ```
   alpha 0xcb40,0x001e,0x0050
   ```

Using `readBMFcSubAll`, 0 (automatic cutoff frequency is enabled) is indicated for each subwoofer channel group: LFE and Subwoofer:

```
alpha 0xce40,0x2804,0x0000,0x0000
```

Using `readBMFcLFE`, 0 (automatic cutoff frequency is enabled) is indicated for LFE channel:

```
alpha 0xcb40,0x0028,0x0000
```

Using `readBMFcSubw`, 0 (automatic cutoff frequency is enabled) is indicated for Subwoofer channel:

```
alpha 0xcb40,0x002a,0x0000
```

2. Change BM Filter Cutoff Control Registers for Main channels cutoff at 40 Hz, Center channel cutoff at 108 Hz, Surround channels cutoff at 120 Hz and Back channel(s) cutoff at 200 Hz:

   To write all four satellite control registers at once, use:

   ```
   writeBMFcSatAllN(40,108,120,200)
                     or
   writeBMFcSatAllN(0x28,0x6c,0x78,0xc8)
   ```

   To write the four satellite control registers separately, use:

   ```
   writeBMFcMainN(40)   or writeBMFcMainN(0x28)
   writeBMFcCntrN(108)  or writeBMFcCntrN(0x6c)
   writeBMFcSurrN(120)  or writeBMFcSurrN(0x78)
   writeBMFcBackN(200)  or writeBMFcBackN(0xc8)
   ```

   Set the LFE filter cutoff control register to 100 Hz. (We'll leave the Subwoofer filter cutoff control register set to 0 for automatic setting). To write the LFEfilter cutoff control register, use:

   ```
   writeBMFcLFEN(100)   or writeBMFcLFEN(0x64)
   ```

While Bass Management is enabled and active, i.e. audio decoding is in progress, read BM Filter Cutoff Status Registers to verify filter cutoff frequencies are set as intended by step 2 above:

Using `readBMFcStatusSatAll`, the returned values indicate Main cutoff is 40 Hz, Center cutoff is 100 Hz (rounded down from selected 108 Hz), Surround cutoff is 120 Hz and Back cutoff is 200 Hz:

```
alpha 0xce40,0x1808,0x0028,0x0064,0x0078,0x00c8
```

Using `readBMFcStatusMain`, the returned value indicates Main cutoff is 40 Hz:

```
alpha 0xcb40,0x0018,0x0028
```

Using `readBMFcStatusCntr`, the returned value indicates Center cutoff is 100 Hz. Note that 108 Hz was quantized (rounded down) to 100 Hz:

```
alpha 0xcb40,0x001a,0x0064
```

Using `readBMFcStatusSurr`, the returned value indicates Surround cutoff is 120 Hz:
```
alpha 0xcb40,0x001c,0x0078
```

Using `readBMFcStatusBack`, the returned value indicates Back cutoff is 200 Hz:
```
alpha 0xcb40,0x001e,0x00c8
```

As described in the “LFE and Subwoofer Filter Cutoff” note above, the Subwoofer filter cutoff for this example is set automatically to the maximum filter cutoff value of all “Small” satellite channels and LFE channel. Using `readBMFcStatusSubAll`, the returned values indicate LFE and Subwoofer filter cutoffs are set to 100 Hz and 200 Hz respectively, with the latter determined by the satellite Back channels having the maximum cutoff frequency:
```
alpha 0xce40,0x2804,0x0064,0x00c8
```

H.8 Bass Management Processing

Bass Management processing, as implemented by BM2, consists of the following sequence. Not all blocks in the following sequence are necessarily performed, based on BM command settings and the audio stream’s requested channel configuration.

H.8.1 Initialize / Reinitialize

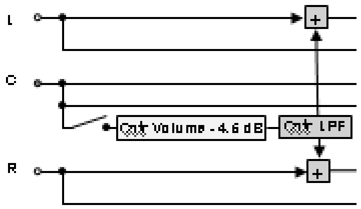
This block is performed under a number of conditions, including:

1. sample rate change
2. speaker configuration change
3. filter cutoff frequency change
4. BM mode change

All filter coefficients and states are updated.

H.8.2 Center bass redirection to Left and Right

This block performs the functions indicated in the upper left in the figure in Section H-1 (BM2 Block Diagram)



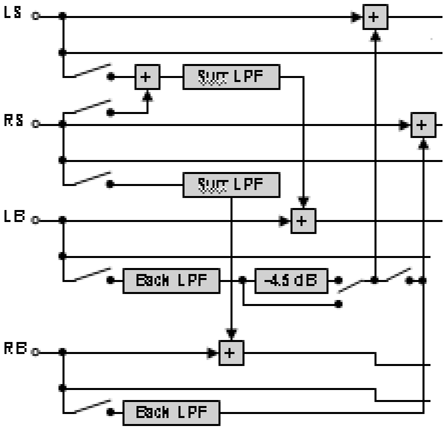
It operates if DOC3 is selected or Local Bass is enabled with Local Bass Front selected, except that if DOC3 is selected and Local Bass is enabled but Local Bass Front is *not* selected, the block is skipped.

Further, the block operates only if DOC0 is *not* selected, L/R are Large, and C is Small. The following processing is performed:

1. Bass in C is attenuated by 4.5dB plus the CntrVolumeN setting, and then redirected into L and R.
2. C is then high-pass filtered.
3. C then masquerades as “Large” until Section H.8.7 (Optional Subwoofer Filtering) in this sequence.

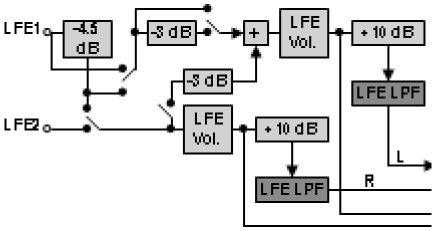
H.8.3 Local Bass Rear

Bass is redirected between Ls/Rs and Lb/Rb if Local Bass is enabled, Local Bass Rear is enabled, and either Ls/Rs are Small with Lb/Rb Large or the opposite. Source channels are high-pass filtered, then masquerade as “Large” until Section H.8.7 (Optional Subwoofer Filtering) in this sequence.



H.8.4 LFE Scaling

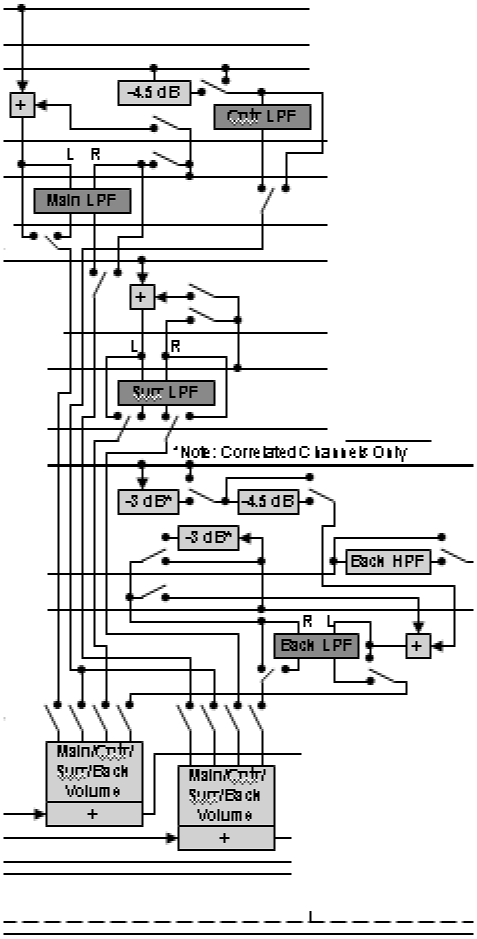
LFE is scaled and, if Digital-only and not DOC0, low-pass filter.



H.8.5 Collect LFE plus Bass from Small channels

The following sequence is performed:

- Add into the scaled LFE channel(s) the bass from “Small” channels[63]. Bass is not collected from “Large” channels, nor from channels masquerading as “Large”.
- If the Subwoofer is present and subwoofer reinforcement is disabled, send the final sum to the Subwoofer channel, else hold in a temporary buffer.

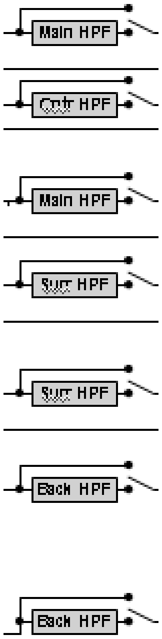


+
-4.5 dB
Cntr LPF
L
R
Main LPF
+
L
R
Surr LPF
*Note: Correlated Channels Only
-3 dB*
-4.5 dB
-3 dB*
Back HPF
R
L
Back LPF
+
Main/Cntr/Surr/Back Volume
+
Main/Cntr/Surr/Back Volume
+
L

[63] The BMOverrideBassCollectionMask register can be used to modify the list of channels from which bass is collected. See Table H.2.5 (Volume Trim)

H.8.6 High-Pass Filter any Small Channels

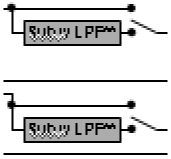
If not DOC0, all Small channels are High-Pass filtered.



H.8.7 Optional Subwoofer Filtering

If external analog circuitry is available (i.e., “BMOCSelectAncillaryAnlgXXXXX”), optionally filter the Subwoofer output (as specified via the BMOCSelectAncillary Subwoofer Low Pass Filter bit).

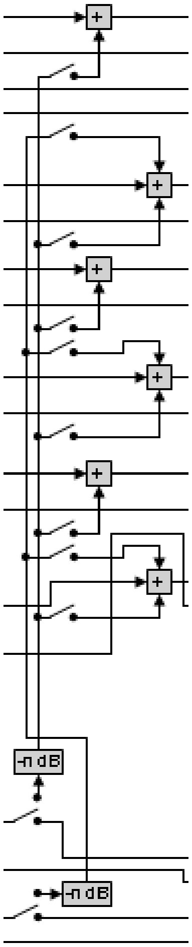
Whether-or-not Subwoofer filtering is selected, this is the end of BM processing for AncillaryAnlg configurations, since external analog circutry performs remaining bass management processing.



H.8.8 End of DOC0 Processing

This is the end-of-the-line if DOC0 is configured.

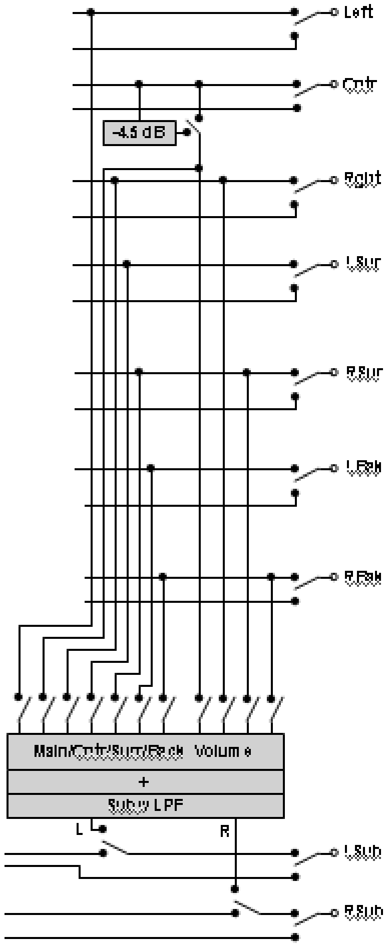
H.8.9 Send Collected Bass to Large channels



This block is performed if no Subwoofer is requested (so that bass must be redirected to Large channels) or Subwoofer Reinforcement is requested. The temporary buffer formed in Section H.8.4 (LFE Scaling) is attenuated by an appropriate amount based on the number of Large channels, and the result is added-into all Large channels.

H.8.10 Copy Bass from Large Channels to Subwoofer

If subwoofer reinforcement has been requested, and subwoofer output has been requested, bass from Large channels is collected, filtered, and sent to the Subwoofer.



APPENDIX I Speaker Location Delay

The Speaker Location Delay (DEL) Algorithm is an Audio Stream Processing (ASP) Algorithm with an Application Interface (API) that permits one to exercise control over various aspects of its operation. This appendix explains the details of this component of the Performance Audio Framework (PA/F) API.

I.1 Control, Status, and Command Registers

Table I-1 (Speaker Location Delay Alpha Code - Using PCE1) gives a list of alpha code symbols for the DEL Algorithm. For an explanation of the various forms of register usage, see Section 7.1 (Register Architecture).

Please keep the following in mind when reading this table:

- Default settings for control registers are shaded. Default settings result at power-up or reset.
- Bit-mapped registers allow the specification of multiple values simultaneously using the logical-or of any combination of the command or response for that register.
- Alpha code of type `write` is used to set values in registers, alpha code of type `read` is used to get values from registers, and alpha code of type `exec` is used to cause more complicated operations to occur.
- Numeric values for the preprocessor symbols shown in this table and appendix are provided in C-language form in the alpha code symbol file `P:\alpha\del_a.h` (when using the DEL Algorithm for Speaker Location Delay) and in `P:\alpha\pce_a.h` (when using the PCE2 Algorithm for Speaker Location Delay, e.g., Feature Set).

The DEL Algorithm has no command registers.

TABLE I-1 Speaker Location Delay Alpha Code - Using PCE1

Register	Alpha Code[a]	Description
Status	readDELStatus	Return entire DEL status structure.
Mode	readDELMode	Indicate if Speaker Location Delay is enabled or disabled.
	writeDELModeDisable	Disable Speaker Location Delay. [b]
	writeDELModeEnable	Enable Speaker Location Delay.
Unit	readDELUnit	Indicate current Speaker Location Delay Unit.
	writeDELUnitTime*Type*	Interpret delay control registers as the time to delay audio data for that channel. The value is expressed according to the *Type*.
	writeDELUnitTimeSamples	The time is expressed as an integral number of samples.
	writeDELUnitTimeMilliseconds	The time is expressed as a 16-bit integer number of milliseconds.
	writeDELUnitTimeMilliseconds2	The time is expressed as a 16-bit value with step size 0.5 of milliseconds.
	writeDELUnitTimeDecimilliseconds	The time is expressed as a 16-bit value in decimilliseconds, that is, as an integral number of tenths of milliseconds.
	writeDELUnitLocation*Type*	Interpret delay control registers as the distance between the speaker associated with that channel and the listener. The value is expressed according to the *Type*.
	writeDELUnitLocationSamples	The distance is expressed as an integral number of samples.
	writeDELUnitLocationCentimeters	The distance is expressed as an integral number of centimeters.
	writeDELUnitLocationFeet	The distance is expressed as an integral number of feet.
	writeDELUnitLocationYards	The distance is expressed as an integral number of yards.
	writeDELUnitLocationMeters	The distance is expressed as an integral number of meters.
Delay Max Channel Number	readDELDelayNumb	Return DEL Maximum Channel Number value.
	writeDELDelayNumbN(NN)	Write the maximum number of channels supported by the DEL algorithm. Currently not supported.
Delay	readDELDelay*Channel*	Return DEL Delay Control Register value for the indicated channel.
	writeDELDelay*Channel*N(*N*)	Select delay for the indicated channel.
Master Delay	readDELDelayMaster	Return the Master Delay Control Register setting.
	writeDELDelayMasterN(0)	Set the Master Delay Control Register to 0
	writeDELDelayMasterN(NN)	Set the Master Delay Control Register to NN. Where NN is an unsigned 16bit value. The delay applied for any channel is equal to the sum of Master Delay and the individual channel delay. Thus, the Master Delay can be used to uniformly delay all channels. The individual channel delay can then be used to apply additional delay for the given channel.

[a] See Table 5-39 (Volume Control Channel Symbols) for a list of symbols appropriate for use to replace the meta-symbol *Channel*.

[b] Once disabled, speaker location delay cannot be re-enabled.

When using PCE2 (e.g., with Feature Set), the alpha commands described in Table I-2 (Speaker Location Delay Alpha Code - Using PCE2) must be used instead for controlling the Speaker Location Delay (DEL) Algorithm.

TABLE I-2 **Speaker Location Delay Alpha Code - Using PCE2**

Register	Alpha Code[a]	Description
Status	readPCEDELStatus	Return entire PCE2 + DEL status structure.
Mode	readDPCEELMode	Indicate if Speaker Location Delay is enabled or disabled.
	writePCEDELModeDisable	Disable Speaker Location Delay. [b]
	writePCEDELModeEnable	Enable Speaker Location Delay.
Unit	readPCEDELUnit	Indicate current Speaker Location Delay Unit.
	writePCEDELUnitTime*Type*	Interpret delay control registers as the time to delay audio data for that channel. The value is expressed according to the *Type*.
	writePCEDELUnitTimeSamples	The time is expressed as an integral number of samples.
	writePCEDELUnitTimeMilliseconds	The time is expressed as a 16-bit integer number of milliseconds.
	writePCEDELUnitTimeMilliseconds2	The time is expressed as a 16-bit value with step size 0.5 of milliseconds.
	writePCEDELUnitTimeDecimilliseconds	The time is expressed as a 16-bit value in decimilliseconds, that is, as an integral number of tenths of milliseconds.
	writePCEDELUnitLocation*Type*	Interpret delay control registers as the distance between the speaker associated with that channel and the listener. The value is expressed according to the *Type*.
	writePCEDELUnitLocationSamples	The distance is expressed as an integral number of samples.
	writePCEDELUnitLocationCentimeters	The distance is expressed as an integral number of centimeters.
	writePCEDELUnitLocationFeet	The distance is expressed as an integral number of feet.
	writePCEDELUnitLocationYards	The distance is expressed as an integral number of yards.
	writePCEDELUnitLocationMeters	The distance is expressed as an integral number of meters.
Delay Max Channel Number	readPCEDELDelayNumb	Return DEL Maximum Channel Number value.
	writePCEDELDelayNumbXX(XX)	Write the maximum number of channels supported by the DEL algorithm. Currently not supported.
Delay	readPCEDELDelay*Channel*	Return DEL Delay Control Register value for the indicated channel.
	writePCEDELDelay*Channel*N(*N*)	Select delay for the indicated channel.
Master Delay	readPCEDELDelayMaster	Return the Master Delay Control Register setting.
	writePCEDELDelayMasterN(0)	Set the Master Delay Control Register to 0
	writePCEDELDelayMasterN(NN)	Set the Master Delay Control Register to NN. Where NN is an unsigned 16bit value. The delay applied for any channel is equal to the sum of Master Delay and the individual channel delay. Thus, the Master Delay can be used to uniformly delay all channels. The individual channel delay can then be used to apply additional delay for the given channel.

[a] See Table 5-39 (Volume Control Channel Symbols) for a list of symbols appropriate for use to replace the meta-symbol *Channel*.

[b] Once disabled, speaker location delay cannot be re-enabled.

I.2 Mode

The DEL Mode Control Register controls basic operation of the DEL Algorithm. If zero, operation is disabled, and if non-zero, operation is enabled.

A special care should be taken to change the mode of the DEL Algorithm or to change the delay value of required channels because runtime change of channel delay value may result in undesired system behavior. Follow the below steps when mode of DEL Algorithm is to be changed or the dealy value of a channel to be changed:

- Set the input to *None* (e.g. send PA17*InNone) for the Audio Zone where DEL algorithm is operating. It will make sure that audio is not processed when control commands are sent for the above changes.
- Send control commands to change the mode or to change the delay value of the required channels
- Set the input to the desired active mode (e.g. for digital input, send PA17*InDigital)

I.3 Units

The DEL Unit Control Register selects the units of delay as described below and as shown in Table I-3 (Speaker Location Delay Units):

- If units of *time* are selected, the value specified in the Master Delay Control register is converted from the particular time-based units to samples according to the sample rate used. Also, the DEL Delay Control Register is converted from the particular time-based units to samples according to the sample rate. The sum of the delay specified by the Master Delay Control register and the DEL Delay Control Register is used as the delay for a given channel. Note that units of "samples" are independent of the sample rate, but all other units are not.
- If units of *location* are selected, the set of values specified in the DEL Delay Control Registers indicate the distance between the speaker for that channel and the listener. These distance values in the DEL Delay Control Registers are used to compute sample-based delays for each channel according to the sample rate in use. The sample-based delay is calculated such that the generated audio output for each channel reaches the listener at the same time. Thus, for example, if the distance specified for the Left speaker is 10 feet, and for the Center speaker is 5 feet, the actual delay that will be applied to the Left and Center channels would be such that the Center output is heard at the same time as the Left output. Thus, the Center will be delayed by 5 feet (converted to the equivalent number of samples) and the Left will not be delayed at all. Note that units of "samples" are independent of the sample rate, but all other units are not.

TABLE I-3 **Speaker Location Delay Units**

Unit	Description
TimeSamples or LocationSamples	Time or distance is specified directly in units of samples.
TimeMilliseconds	Time is specified in units of milliseconds.
TimeMilliseconds2	Time is specified in units of one-half milliseconds.
LocationCentimeters	Distance is specified in units of centimeters.
LocationFeet	Distance is specified in units of feet.
LocationYards	Distance is specified in units of yards.
LocationMeters	Distance is specified in units of meters.
TimeDecimilliseconds	Time is specified in units of decimilliseconds.

I.4 Operation

The operation of speaker location delay is described in this section.

I.4.1 Default Operation

By default, speaker location delay is enabled for time-based operation, with delay expressed in milliseconds. The delays are set to 0 ms on the Center Channel, 5 ms on the Left and Right Surround Channels, and 5 ms on the Left and Right Back Channels.

Feature Set: Delay is supported as shown in Table I-4 (Dolby Delay Support), as per Dolby requirements.

Feature Set : Delay is supported as shown in Table I-5 (THX Delay Support), as per THX requirements.

These tables show both the default value of the delay for each channel, as well as the range of delay values possible.

TABLE I-4 **Dolby Delay Support**

Channel	Default Delay	Delay Range (at 96 kHz)
Left	0 ms	0 (none) ms
Cntr	0 ms	0 - 5 ms
Rght	0 ms	0 (none) ms
LSur	5 ms	0 - 15 ms
RSur	5 ms	0 - 15 ms
LBak	5 ms	0 - 15 ms

Channel	Default Delay	Delay Range (at 96 kHz)
RBak	5 ms	0 - 15 ms
Subw	0 ms	0 (none) ms

TABLE I-5 **THX Delay Support**

Channel	Default Delay	Delay Range (at 96 kHz)
Left	0 ms	0 - 10 ms
Cntr	0 ms	0 - 10 ms
Rght	0 ms	0 - 10 ms
LSur	5 ms	0 - 20 ms
RSur	5 ms	0 - 20 ms
LBak	5 ms	0 - 20 ms
RBak	5 ms	0 - 20 ms
Subw	0 ms	0 - 10 ms

Note: ***Setting Delay Values Beyond the Maximum for Feature Set***

This note is not applicable ***to PA feature sets that include THX, such as Feature Set .***

As explained in Section I.4.4 (Operational Limitations), setting values outside the maximum delay specified in Figure I-4 (Dolby Delay Support) have no effect.

For example, Figure I-4 (Dolby Delay Support) shows that no delay capability is provided for the Left, Right, and Subwoofer channels. Keep in mind that it is possible to write delay values for these channels using alpha code `writeDELDelay`*`Channel`*`N(N)`*, where* *`Channel`* *is either* `Left`, `Rght` *or* `Subw`*. One may also read back these same values using alpha code* `readDELDelay`*`Channel`*. *However, no delay capability is provided for these channels, and the delay values written will have no effect.*

I.4.2 Time-Based Operation

The total delay that needs to be applied for each channel is the sum of the Master Delay Control register value and the DEL Delay control register value for given channel. The calculated time-based values are converted directly to a delay in samples for each channel according to the units in use as described in Section Section I.3 (Units). These delay values in samples are then used to delay the audio data for that channel.

I.4.3 Distance-Based Operation

The distance-based values given in the DEL Delay Control Registers are converted indirectly to a delay in samples for each channel according to the units in use as described in Section I.3 (Units). These delay values in samples are then used to delay the audio data for that channel. Please note that the Master Delay Control register value has no impact in the distance-based operation.

I.4.4 Operational Limitations

The delay lines for a given instantiation of the Performance Audio Framework are of a limited length. This limited length is the maximum delay possible as given in the Delay Range column of Figure I-4 (Dolby Delay Support) for non-THX systems, or Table I-5 (THX Delay Support) for THX systems. Any attempt to specify a delay of greater than that which can be accommodated by this limited length will be saturated to the greatest value which can be supported by that instantiation for that channel. In the extreme case, some channels in some instantiations support no delay, and even though delay may be specified for that channel using either time- or distance-based quantities, no delay will be supplied for that channel. Please note for the time-based operation total delay that is applied for each channel is the sum of the Master Delay Control register value and the DEL Delay control register value for given channel. Thus, care should be taken to ensure that the total delay requested for any channel is not greater than what can be accommodated by the limited length of the delay line.

The granularity of the delay in the implementation of Speaker Location Delay is a single audio data sample—no interpolation of audio data is performed to achieve delay in a granularity of less that a single audio data sample.

I.4.5 Operational Features

The implementation provided for Speaker Location Delay automatically “clears the delay line” upon start-up. This is achieved not by writing zeros to the memory but by separating the implementation into two phases, “start-up” and “continuous.” In the start-up phase, audio data is written to the delay line but the result from the delay line is always zero-valued audio data. In the continuous phase, audio data is actually both written to and read back from the delay line as appropriate.

I.4.6 Usage of Audio Frame Data Structure Registers

A pointer to the Audio Frame Data Structure is passed to an ASP Algorithm as part of both the reset and apply function. It encapsulates a “frame” of audio sample data on which processing is to take place.

Table I-6 (Usage of Audio Frame Data Structure Quantities) illustrates the registers in the Audio Frame Data Structure which are utilized by the DEL Algorithm and whether the registers are read, written, or read *and* written.

TABLE I-6 **Usage of Audio Frame Data Structure Quantities**

Audio Frame Data Structure Register	Read/Written
channelConfigurationStream	R
sampleRate	R
sampleCount	R
data.sample	R/W
sampleProcess	R/W

Determine whether delay requested is time or location based.

For time-based delay requests, the delay values requested are propagated to the internal DEL pConfig structure for channels available in the mask of the pConfig structure.

For location-based delay requests, the delay values requested are propagated to the internal DEL pConfig structure, for channels available in the mask of the pConfig structure by subtracting the delay values from the maximum delay value requested from the status structure.

The internal delay routine is then called for channels in the input stream and also available in the mask of the pConfig structure. The amount of delay applied is denoted in the internal DEL pConfig structure, whose values were computed in previous steps.

The DEL field of the Sample Process register is set when DEL processing has successfully occured.

APPENDIX J MIPS Load Algorithm (ML0)

The MIPS Load (ML0) Algorithm is an Audio Stream Processing (ASP) Algorithm with an Application Interface (API) that permits one to exercise control over its operation. This appendix explains the details of this component of the Performance Audio Framework (PA/F) API.

J.1 Control, Status, and Command Registers

Table J-1 (MIPS Load Alpha Code) gives a list of alpha code symbols for the ML0 Algorithm.

Please keep the following in mind when reading this table:

- Default settings for control registers are shaded. Default settings result at power-up or reset.
- Alpha code of type `write` is used to set values in registers and alpha code of type `read` is used to get values from registers.
- Numeric values for the preprocessor symbols shown in this table and appendix are provided in C-language form in the alpha code symbol file `P:\alpha\ml_a.h`.

The ML0 Algorithm has no command registers.

TABLE J-1 **MIPS Load Alpha Code**

Register	Alpha Code	Description
Status	readMLStatus	Standard ASP alpha code for returning entire ML status structure.
Control	readMLControl	Identical to readMLStatus.

Register	Alpha Code	Description
Mode	readMLMode	Return ML Mode Control Register value.
	writeMLModeDisable	Disable MIPS Load.
	writeMLModeEnable	Enable MIPS Load.
Count	readMLCount	Return ML Count Register value.
	writeMLCountN(*NN*)	Write ML Count Register with value *NN*, where *NN* is a 16-bit unsigned integer.
	writeMLCountN(0x0)	Write ML Count Register with a value of zero. While MIPS Load is enabled (writeMLModeEnable), a zero value causes MIPS Load processing to be bypassed.

J.2 Mode

The ML Mode Control Register controls basic operation of the ML0 Algorithm. If zero, operation is disabled, and if non-zero, operation is enabled.

J.3 Count

The ML Count Register controls amount of MIPS Load added to Audio Stream Processing. The purpose of adding MIPS Load (CPU load) is to allow forcing the DSP into a “MIPS overload” condition for evaluation of system response/recovery.

MIPS Load Demonstration Algorithm processing performs no modification of audio sample data. MIPS Load is added through repetitive execution of a null routine based on the ML Count Register setting. The default setting of zero effectively bypasses this processing.

As the goal of adding MIPS Load is to simply achieve an overload condition, a simple loop has been implemented which uses the ML Count Register value as a multiplier for loop iterations per audio data sample. Sufficient range is provided for producing an overload condition during operation of available decoders. The only quantification given to Count is that increasing its value increases MIPS Load per audio sample. Given that CPU operating speed may vary and other variables, this implementation suffices for achieving MIPS overload and minimizes memory usage for what is essentially a test tool.

Number of iterations is independent of audio channel configuration, i.e. MIPS Load is effectively added per audio data sample of one channel to maintain consistent MIPS Load across various channel configurations

The Texas Instruments (TI) Code Composer Studio “CPU Load Graph” feature can be used to monitor the MIPS Load added by this processing.

APPENDIX K Synchronous Rate Conversion Number 4

The Synchronous Rate Conversion Algorithm is an Audio Stream Processing (ASP) Algorithm with an Application Interface (API) that permits one to exercise control over various aspects of its operation. This appendix explains the details of this component of the Performance Audio Framework (PA/F) API.

The Synchronous Rate Conversion Number 4 (SRC4) Algorithm provides a limited but powerful form of sample rate conversion suitable for use with PA. In particular, it provides synchronous sample rate conversion to produce output that is quadruple rate, double rate, full rate, half rate, or quarter rate relative to the input to this algorithm. Other (synchronous) ratios and asynchronous sample rate conversion are not available with this algorithm.

The Synchronous Rate Conversion Number 4 (SRC4) Algorithm may be instantiated multiple times within a single implementation of the Performance Audio Framework. The SRC4 Algorithm may be available within multiple audio streams. The SRC4 Algorithm may be present at both the head and the tail of the ASP Chain associated with a particular audio stream:

- At the head of an audio stream, full, half, or quarter rate output may be produced for one to eight channel input. This is referred to as Downsampling.
- At the tail of the audio stream, full, double, or quadruple rate output may be produced for PCM, and for bit-stream input under some condition. This is referred to as Upsampling.

K.1 Control, Status, and Command Registers

Table K-1 (Synchronous Rate Conversion Alpha Code) gives a list of alpha code symbols for the SRC4 Algorithm. For an explanation of the various forms of register usage, see Section 4.1 (Register Architecture).

Please keep the following in mind when reading this table:

- Default settings for control registers are shaded. Default settings result at power-up or reset.
- Alpha code of types `write` and `read` are used to set and get values to and from registers, respectively. Alpha codes of type `wrote` indicate read-responses of status registers (i.e., registers that are not writable).
- Numeric values for the preprocessor symbols shown in this table and appendix are provided in C-language form in the alpha code symbol file `P:\alpha\src_a.h`.
- As described above, there may be multiple instantiations of the SRC4 Algorithm: multiple instantiations in multiple audio streams, and multiple instantiations within a single audio stream.

  Because there are multiple instantiations of the SRC4 Algorithm in multiple audio streams, alpha codes must be directed at the appropriate audio stream using tools and techniques as described in Chapter 4 (Application Interface - Overview).

Note: Within a single audio stream, alternative forms of the alpha code symbols are used to communicate with the appropriate instantiation. For the alpha code symbols given in Table K-1 (Synchronous Rate Conversion Alpha Code), the notation "SRC" is to be replaced by "SRC_A_" for access to the first SRC4 Algorithm Instantiation in the audio stream, and by "SRC_B_" for access to the second SRC4 Algorithm Instantiation in the audio stream.

The SRC4 Algorithm has no command registers.

TABLE K-1 **Synchronous Rate Conversion Alpha Code**

Register	Alpha Code[a]	Description
Status	readSRCStatus	Return entire SRC status structure.
Control	readSRCControl	Return control registers of SRC status structure.
Mode	readSRCMode	Indicate if Synchronous Rate Conversion is enabled or disabled.
	writeSRCModeDisable	Disable Synchronous Rate Conversion.
	writeSRCModeEnable	Enable Synchronous Rate Conversion.

Register	Alpha Code[a]	Description
Rate Request[b]	readSRCRateRequest	Return what the current rate request is.
	writeSRCRateRequestFull	Select full-rate output.
	writeSRCRateRequestHalf	Select half-rate output.
	writeSRCRateRequestQuarter	Select quarter-rate output.
	writeSRCRateRequestDouble	Select double-rate output.
	writeSRCRateRequestQuadruple	Select quadruple-rate output.
	writeSRCRateRequestMax192	Select automatic determination of output rate so that the maximum output rate is 192kHz.
	writeSRCRateRequestMax96	Select automatic determination of output rate so that the maximum output rate is 96kHz.
	writeSRCRateRequestMax48	Select automatic determination of output rate so that the maximum output rate is 48kHz.
	writeSRCRateRequestMin128	Select automatic determination of output rate so that the minimum output rate is 128 kHz.
	writeSRCRateRequestMin64	Select automatic determination of output rate so that the minimum output rate is 64 kHz.
	writeSRCRateRequestMin32	Select automatic determination of output rate so that the minimum output rate is 32 kHz.
Rate Stream	readSRCRateStream	Return intended sample rate ratio.
	wroteSRCRateStreamFull	Intended output is full-rate.
	wroteSRCRateStreamHalf	Intended output is half-rate.
	wroteSRCRateStreamQuarter	Intended output is quarter-rate.
	wroteSRCRateStreamDouble	Intended output is double-rate.
	wroteSRCRateStreamQuadruple	Intended output is quadruple-rate.
Sample Rate	readSRCSampleRate	Indicate actual output sample rate.
	wroteSRCSampleRateNone	If the SRC4 Algorithm for some reason is not able to perform the sample rate conversion that is requested, the SRCSampleRate status variable is set to None.

[a] The notation “SRC” here is to be replaced by “SRC_A_” for access to the first instantiation in the ASP Chain and by “SRC_B_” for access to the second instantiation in the ASP Chain when multiple instantiations are present in a single audio stream.

[b] Selection of the *rate* is a deferred control operation as described in Section K.1.2 (Rate Ratio). In certain cases, it may be necessary to control the percentage of the audio frame buffer utilized by the Decode Algorithm as described in Section K.2.3 (Upsampling Considerations).

K.1.1 Mode

The SRC Mode Control Register controls basic operation of the SRC4 Algorithm. If zero, operation is disabled, and if non-zero, operation is enabled.

K.1.2 Rate Ratio

The term *rate ratio* is used here to describe the ratio between the sample rate of the audio data at the input to the SRC4 Algorithm and that intended for the audio data at the output of the SRC4 Algorithm:

- If SRC4 is used to Upsample, it is able to yield a sample rate increase with a ratio of 1:2 or 1:4.
- If SRC4 is used to Downsample, it is able to yield a sample rate decrease with a ratio of 2:1 or 4:1.

The SRC Rate Request Control Register selects the desired rate ratio in one of two manners:

- A rate ratio may be *directly requested*. If a rate ratio is directly selected, that rate ratio is used.
- A rate ratio may be *indirectly requested*. If a minimum sample rate is selected, a rate ratio is determined that is appropriate to produce output with a sample rate that is no lower than that specified. Likewise, if a maximum sample rate is selected, a rate ratio is determined that is appropriate to produce output with a sample rate that is no higher than that specified.

If a specific sample rate is desired, requesting a rate ratio indirectly is the preferred method. The indirect method prevents the possibility of a rate ratio creating an unsupported sample rate and potentially causing undesirable audio effects.

The SRC Rate Stream Status Register reports the sample rate conversion intended by the algorithm. This is the rate ratio as interpreted by the algorithm. This value indicates only the intended sample rate conversion, however, such conversion may or may not be possible in the current audio processing state.

The SRC Sample Rate Status register reports the actual sample rate at the output of the algorithm if Synchronous Rate Conversion is active. If Synchronous Rate Conversion is not active, this status register reports the sample rate None.

IMPORTANT:Selection of the Rate Ratio is a Deferred Control Operation

Selection of the rate ratio is a deferred control operation requiring additional action for it to be resolved ultimately into operational sample rate conversion. Resolution is accomplished by resetting Decode Processing as described in Section 7.4.3 (Overcoming Control Operation Deferral). The rate ratio and consequently the sample rate will NOT change until Decode Processing is reset.

IMPORTANT:Selection of an Upsampling Rate Ratio requires a change in Buffer Ratio

Prior to effective operation of an upsampling rate ratio, the buffer ratio must be changed in order to accommodate the request. The buffer ratio is part of Audio Decoding, and is described in Section K.2.3 (Upsampling Considerations).

K.2 Operation

The operation of Synchronous Rate Conversion in PA as implemented by the SRC4 Algorithm is described in this section.

K.2.1 Instantiation of SRC4

The Synchronous Rate Conversion Number 4 (SRC4) Algorithm is instantiated through the use of initialization parameters. These initialization parameters consist of information that is used by the SRC4 algorithm to make certain decisions about the number of channels, the maximum or minimum sample rate, filter coefficients, and allowable Upsampling or Downsampling capability of an instance of SRC4. Detail about these parameters is provided in this section.

When instantiating the SRC4 Algorithm, the parameters can be chosen that best match the requirement. There are default parameters provided to satisfy various requirements (see Table K-2 (SRC Parameters description) for more details). For example, if the system is constrained such that the input will never exceed two-channels into the SRC4 instance and the requirement is that the output sample rate must never exceed 48 kHz, the parameter ISRC_PARAMS_MAX48 should be used. Likewise, if the system is constrained such that the SRC4 instance can receive eight-channels and the requirement is to Downsample, the parameter ISRC_PARAMS_DS_8CH can be used.

Please note that the memory allocated by the SRC4 Algorithm may vary based on the parameter used to initialize the algorithm. For example, ISRC_PARAMS_DS_2CH allocates less memory than ISRC_PARAMS_DS_8CH. It should also be noted that ISRC_PARAMS_US_2CH allocates less memory than ISRC_PARAMS_DS_2CH (See the `ISRC_memRec` definitions in `isrc.c` and `isrc_hbw.c` for more details). Thus, care should be taken when choosing the parameters to initialize the SRC4 Algorithm.

TABLE K-2 **SRC Parameters description**

SRC Parameters	Functionality
ISRC_PARAMS_MAX192	Allow for maximum frequency to be 192 kHz. Both upsampling up to 4X and downsampling down to 4:1 ratio on 8 channel are supported.
ISRC_PARAMS_MAX192_HBW	Same as ISRC_PARAMS_MAX192, but uses High Bandwidth filter. See Section K.2.7 (SRC Filters)
ISRC_PARAMS_MAX48	Allow for maximum frequency to be 48 kHz. If the sampling frequency goes beyond 48 kHz, downsampling is performed. Max 2 channel downsampling down to 4:1 ratio is supported.
ISRC_PARAMS_MAX48_HBW	Same as ISRC_PARAMS_MAX48, but uses High Bandwidth filter. See Section K.2.7 (SRC Filters)
ISRC_PARAMS_MIN32	Allow for minimum frequency to be 32 kHz. If the sampling frequency goes below 32 kHz, upsampling is performed. Max 8 channel upsampling pot 4X is supported.
ISRC_PARAMS_MIN32_HBW	Same as ISRC_PARAMS_MIN32, but uses High Bandwidth filter. See Section K.2.7 (SRC Filters)
ISRC_PARAMS_DS_2CH	Allow for maximum 2 channel downsampling down to 4:1 ratio. Downsampling is not performed automatically for any sample rate. The downsampling has to be requested specifically.
ISRC_PARAMS_DS_2CH_HBW	Same as ISRC_PARAMS_DS_2CH, but uses High Bandwidth filter. See Section K.2.7 (SRC Filters)
ISRC_PARAMS_DS_8CH	Allow for maximum 8 channel downsampling down to 4:1 ratio. Downsampling is not performed automatically for any sample rate. The downsampling has to be requested specifically.
ISRC_PARAMS_DS_10CH	Allow for maximum 10 channel downsampling down to 4:1 ratio. Downsampling is not performed automatically for any sample rate. The downsampling has to be requested specifically.
ISRC_PARAMS_DS_8CH_HBW	Same as ISRC_PARAMS_DS_8CH, but uses High Bandwidth filter. See Section K.2.7 (SRC Filters)
ISRC_PARAMS_DS_10CH_HBW	Same as ISRC_PARAMS_DS_10CH, but uses High Bandwidth filter. See Section K.2.7 (SRC Filters)
ISRC_PARAMS_US_2CH	Allow for maximum 2 channel upsampling upto 4X. Upsampling is not performed automatically for any sample rate. The upsampling has to be requested specifically.
ISRC_PARAMS_US_2CH_HBW	Same as ISRC_PARAMS_US_2CH, but uses High Bandwidth filter. See Section K.2.7 (SRC Filters)
ISRC_PARAMS_US_8CH	Allow for maximum 8 channel upsampling upto 4X. Upsampling is not performed automatically for any sample rate. The upsampling has to be requested specifically.

SRC Parameters	Functionality
ISRC_PARAMS_US_10CH	Allow for maximum 10 channel upsampling upto 4X. Upsampling is not performed automatically for any sample rate. The upsampling has to be requested specifically.
ISRC_PARAMS_US_8CH_HBW	Same as ISRC_PARAMS_US_8CH, but uses High Bandwidth filter. See Section K.2.7 (SRC Filters)
ISRC_PARAMS_US_10CH_HBW	Same as ISRC_PARAMS_US_10CH, but uses High Bandwidth filter. See Section K.2.7 (SRC Filters)
ISRC_PARAMS	Default params defined to ISRC_PARAMS_MAX192

K.2.1.1 Alternate SRC Parameters

The “High Band Width” filters are included in certain releases of the *PA SDK*. They can be selected at build time by specifying use of ISRC_PARAMS_*XXX*_HBW, in the appropriate `*patchs.c` file. The HBW filters are of higher order and allow 20/40kHz output bandwidth operation at the sample rates multiple of 44.1kHz. (See Table K-5, Table K-6, Table K-7 and Table K-8 for more information). These definitions are provided in the files: `irsc_HBW.c`.

Since the filter length is increased by about 50% for HBW parameters, the computation MIPS and the IRAM requirement for filter states will be increased by 40 ~ 50% from standard parameters to HBW parameters.

Due to the increased MIPS/memory requirement of the HBW parameters, only the standard build Y-Topology, Feature Set 8 build uses HBW parameters, although this can be customized for custom system needs.

K.2.2 Parameter Definition

This section provides more detail on the key components of the SRC4 initialization parameters.

The key SRC4 Initialization Parameter components are defined as follows (referring to `isrc.c` and `isrc_hbw.c`):

- `rateRequest`: the Sample Rate request value (see Section K.1.2 (Rate Ratio)). This value provided via the initialization parameter is the initialization default for that particular instance’s rateRequest register.
- `*dn1toH`: coefficients for 2:1 down-sampling
- `*dnHtoQ`: coefficients for 4:1 down-sampling
- `*up1to2`: coefficients for 1:2 up-sampling
- `*up2to4`: coefficients for 1:4 up-sampling
- `nChannels`: maximum number of channels

TABLE K-3 **SRC4 Initialization Parameter Definitions**

ISRC_Param entry	**Parameter Name**						
	MAX192	**MAX48**	**MIN32**	**DS_2CH**	**DS_8CH**	**US_2CH**	**US_8CH**
rateRequest (default)	Max 192	Max 48	Min 32	Full	Full	Full	Full
*dn1toH (2:1)	Yes	Yes	No	Yes	Yes	No	No
*dnHtoQ (4:1)	Yes	Yes	No	Yes	Yes	No	No
*up1to2 (1:2)	Yes	No	Yes	No	No	Yes	Yes
*up2to4 (1:4)	Yes	No	Yes	No	No	Yes	Yes
nChannels	8	2	8	2	8	2	8

Each instance of SRC makes a decision at run-time on what downsampling/upsampling conversion to perform based on:

- the instantiation parameter
- the *SRCRateRequest* setting
- the number of channels present in the input audio frame
- the sample rate indicated in the audio frame.

In the case wherein the *SRCRateRequest* is a specific factor (i.e. ‘*Double’*, ‘*Quadruple’*, ‘*Half’* or ‘*Quarter’*), the requested conversion is performed if:

- the parameter supports the requested Upsampling/Downsampling capability, **and**
- the parameter supports the number of channels present in the audio frame, **and**
- the resulting output sample rate is supported by PA/F

In the case wherein *SRCRateRequest* is specified in terms of minimum/maximum sample rate, the SRC instance first determines the appropriate conversion ratio between input sample rate and output sample rate as per Section K.1.2 (Rate Ratio), then performs the requested sample rate conversion if the parameter supports that specific conversion ratio.

K.2.3 Upsampling Considerations

The Decode Buffer Ratio Control Register controls the percentage of the audio frame buffer utilized by the Decode Algorithm as described in Section 8.2.8 (Audio Decode Buffer Ratio) and Table 8-24 (Alpha Code for Audio Decode Buffer Ratio). If 100% of the audio frame buffer is utilized by the Decode Algorithm, upsampling cannot be performed by the SRC4 Algorithm. Limitations on its use for this application are noted in Limitations.

To allow the SRC4 Algorithm to be used for 1:2 upsampling, the Decode Buffer Ratio Control Register must be set for 50% or 25% utilization. To allow the SRC4 Algorithm to be used for 1:4 upsampling, the Decode Buffer Ratio Control Register must be set for 25% utilization.

K.2.4 Downsampling Considerations

Synchronous Rate Conversion Number 4 can be used to reduce the sample rate, e.g., for a Digital Record Output (DRO) jack for an A/V Receiver. Limitations on its use for this application are noted in Limitations.

K.2.5 Limitations

Synchronous Rate Conversion Number 4 provides only synchronous sample rate conversion with integral conversion ratios 1:1, 2:1, 4:1, 1:2, and 1:4. *Synchronous* here means that the hardware guarantees that the clocks associated with the input and output are frequency-synchronous, that is, derived from the same base clock signal. Synchronous Rate Conversion Number 4 does not provide the following:

- Synchronous sample rate conversion at rates other than those listed above. In particular, conversion from 48 kHz to 44.1 kHz, or from 44.1kHz to 48kHz, is not supported, even if the clocks are frequency-synchronous with each other.
- Asynchronous sample rate conversion for any ratio of sample rates, even 1:1, where the clocks are not frequency-synchronous with each other.

Not all instantiations of the algorithm for Synchronous Rate Conversion Number 4 provide all conversion ratios. In particular, it is typical that either only downsampling or upsampling is provided by any given instantiation (except with ISRC_PARAMS_MAX192).

Not all instantiations of the algorithm for Synchronous Rate Conversion Number 4 provide processing for all channels. When this is the case, sample rate conversion will not be engaged if it cannot be provided for all channels that are present.

A change in the number of channels to be processed which causes sample rate conversion to be engaged or disengaged due to this limitation may cause audio artifacts in the output.

Synchronous Rate Conversion Number 4 may produce output at sample rates that are not fully supported by other system components. In particular, for PA, the following limitations apply:

- The Downsampling SRC4 Algorithm Instantiation can produce output at sample rates less than 32 kHz. Operation of some system components, in particular Bass Management, is undefined in these cases.

Due to hardware clocking limitations, SRC may be incapable of performing upsampling/downsampling for certain input/output combinations. For example, some hardware implementations do not support the clocking of analog inputs and analog outputs at different rates. In these cases, SRC may indicate (via readSRCRateStream and readSRCSampleRate) that it has successfully upsampled/downsampled, even though it hasn't. See notes in Chapter 2.2.2, “Audio System Connections,” on page 2-30 for more-specific information.

K.2.6 Usage of Audio Frame Data Structure Registers

A pointer to the Audio Frame Data Structure is passed to an ASP Algorithm as part of both the reset and apply function. It encapsulates a “frame” of audio sample data on which processing is to take place.

Table K-4 (Usage of Audio Frame Data Structure Quantities) illustrates the registers in the Audio Frame Data Structure which are utilized by the SRC4 Algorithm and whether the registers are read, written, or read *and* written.

TABLE K-4 **Usage of Audio Frame Data Structure Quantities**

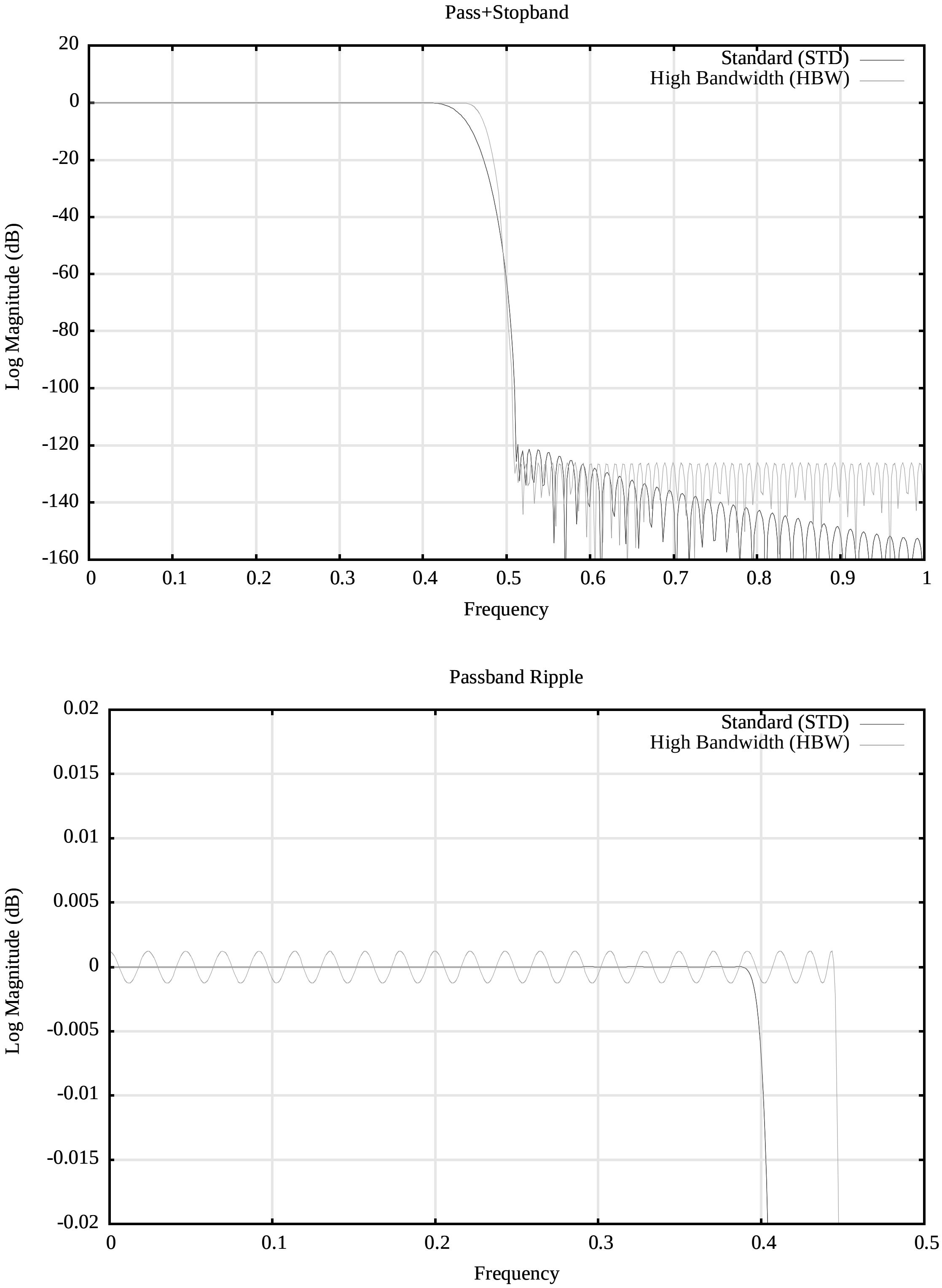
Audio Frame Data Structure Register	Read/Written
channelConfigurationStream	R
sampleRate	R
sampleCount	R
data.sample	R/W
sampleProcess	W

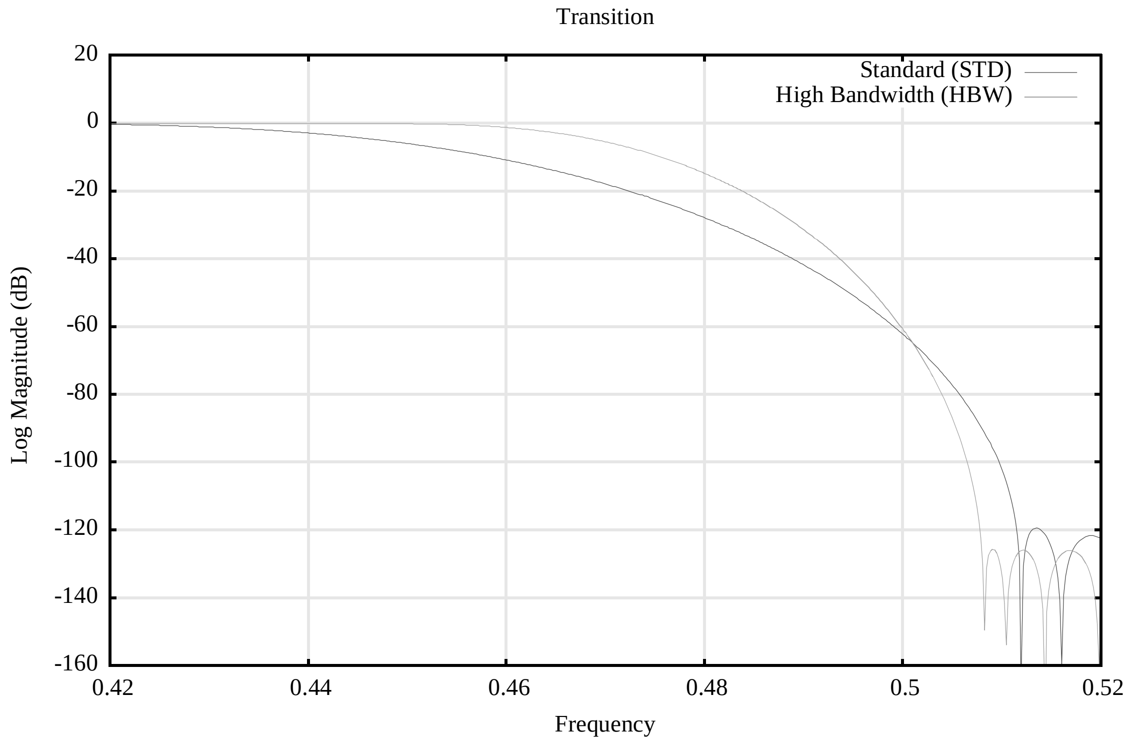
The SRC field of the Sample Process register is set when SRC processing has successfully occurred.

K.2.7 SRC Filters

Figure K-1, **Figure K-2, Figure K-3** and **Figure K-4** show the filters used in up- and down-sampling. “Standard” filters are provided by default in Feature Sets 7 and 9, while “High Bandwidth” filters are provided by default in Feature Set 8.

FIGURE K-1 **Lowpass Filter Used for 2:1 Downsampling and 2nd Stage of 4:1 Downsampling**




Transition
Standard (STD)
High Bandwidth (HBW)
20
0
-20
-40
-60
-80
-100
-120
-140
-160
0.42
0.44
0.46
0.48
0.5
0.52
Log Magnitude (dB)
Frequency

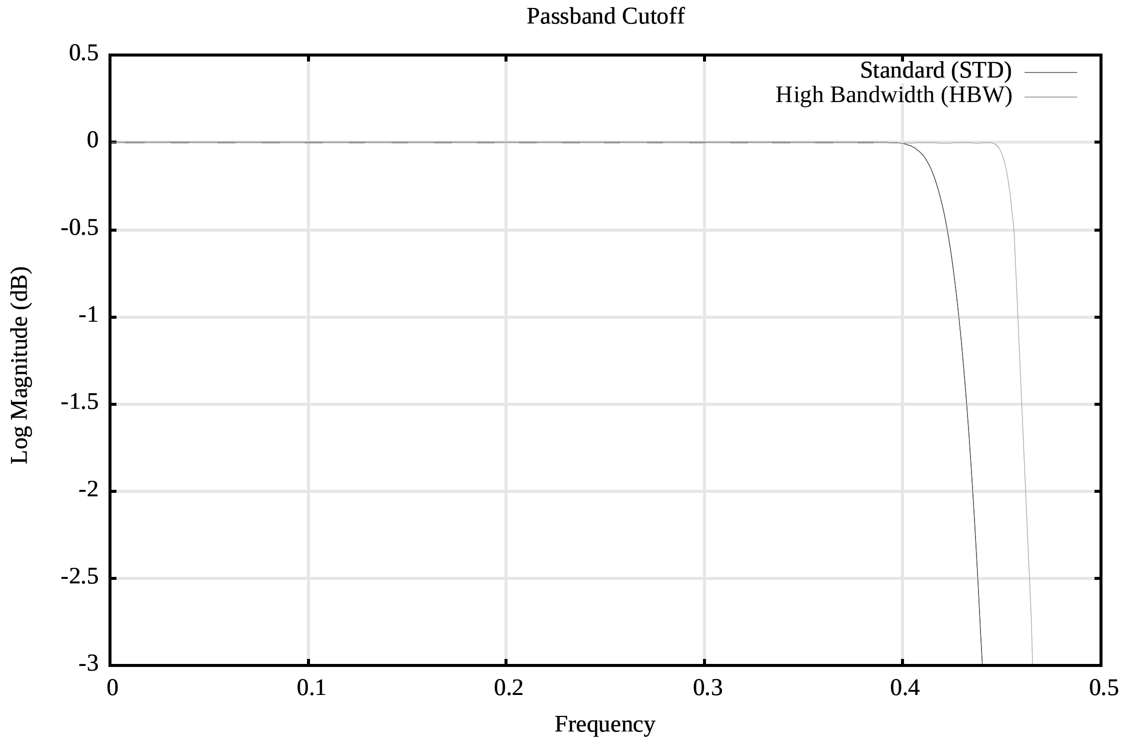
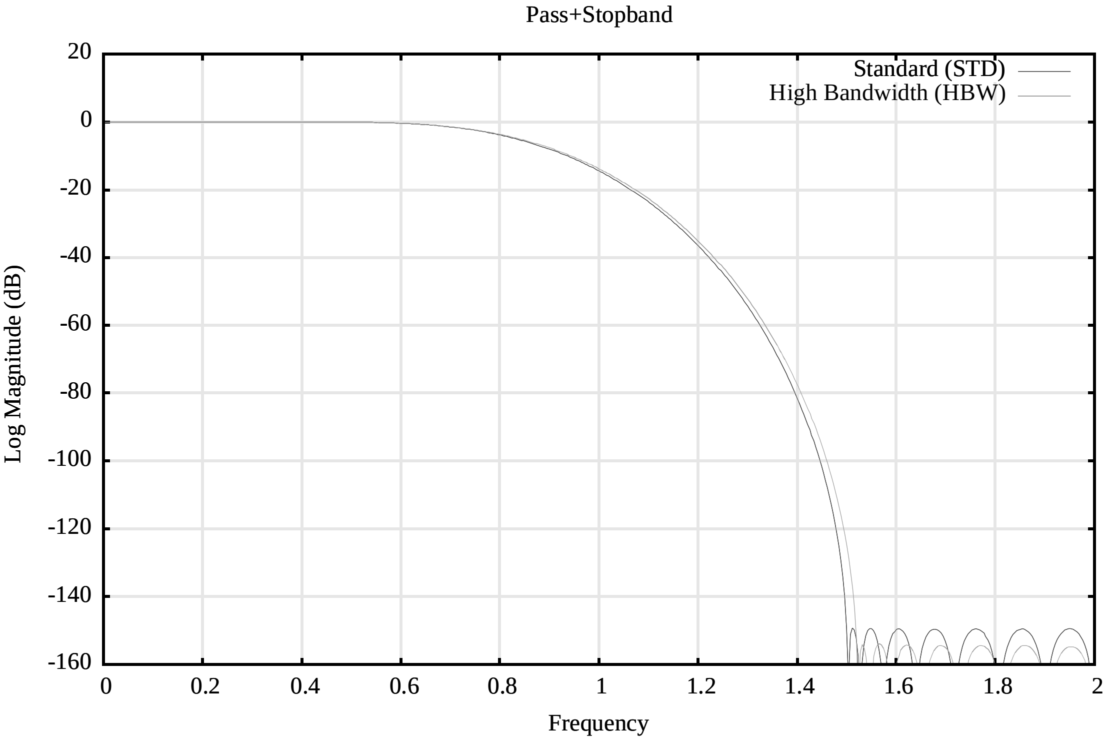

Passband Cutoff
Standard (STD)
High Bandwidth (HBW)
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
0
0.1
0.2
0.3
0.4
0.5
Log Magnitude (dB)
Frequency

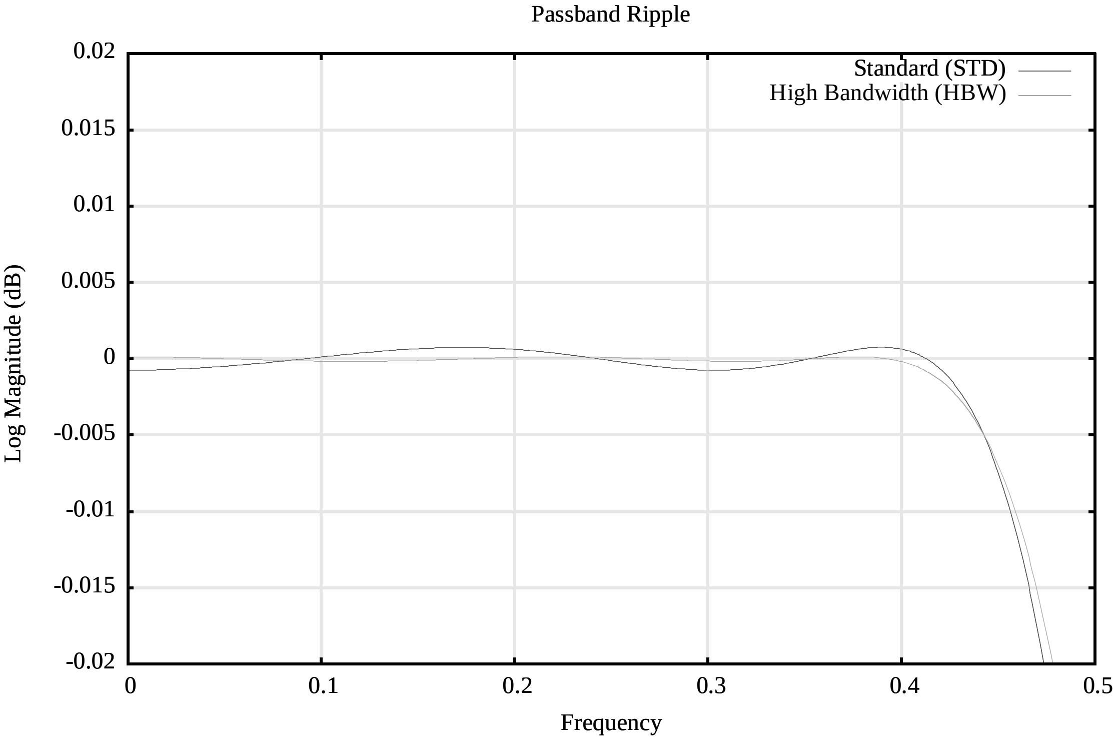
TABLE K-5

Filter Types for Figure K-1

Filter	Type
Standard	128-tap FIR
High Bandwidth	192-tap FIR

FIGURE K-2 **Lowpass Filter Used for 1st stage of 4:1 Downsampling**





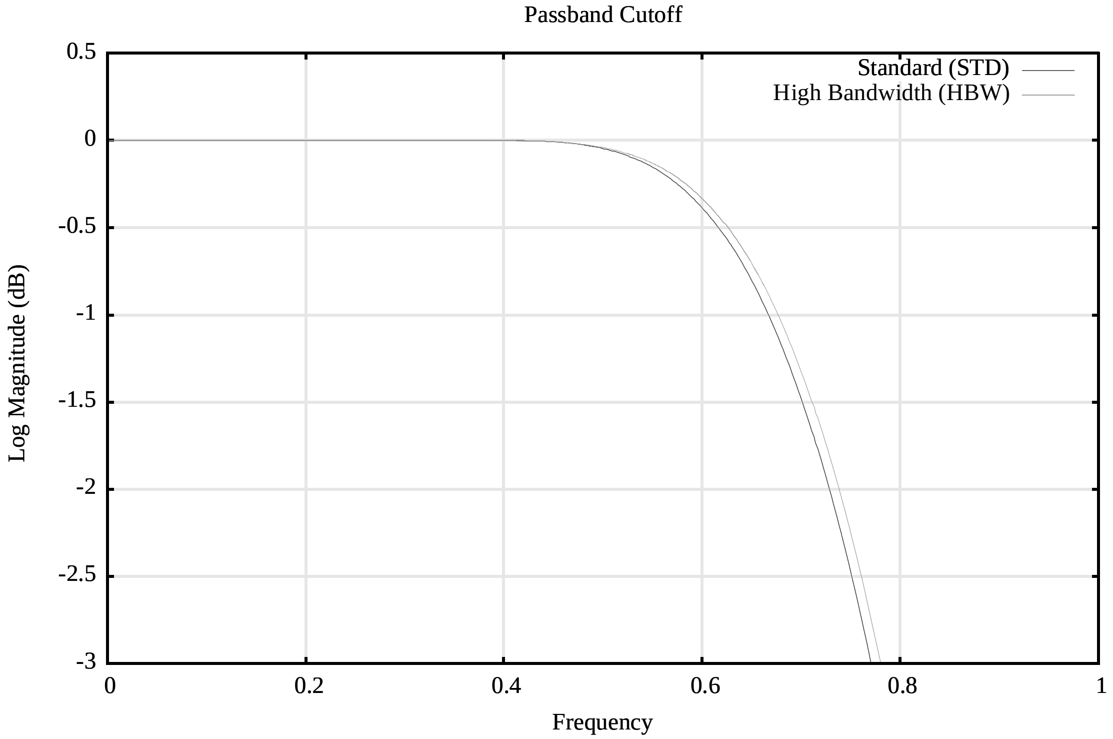
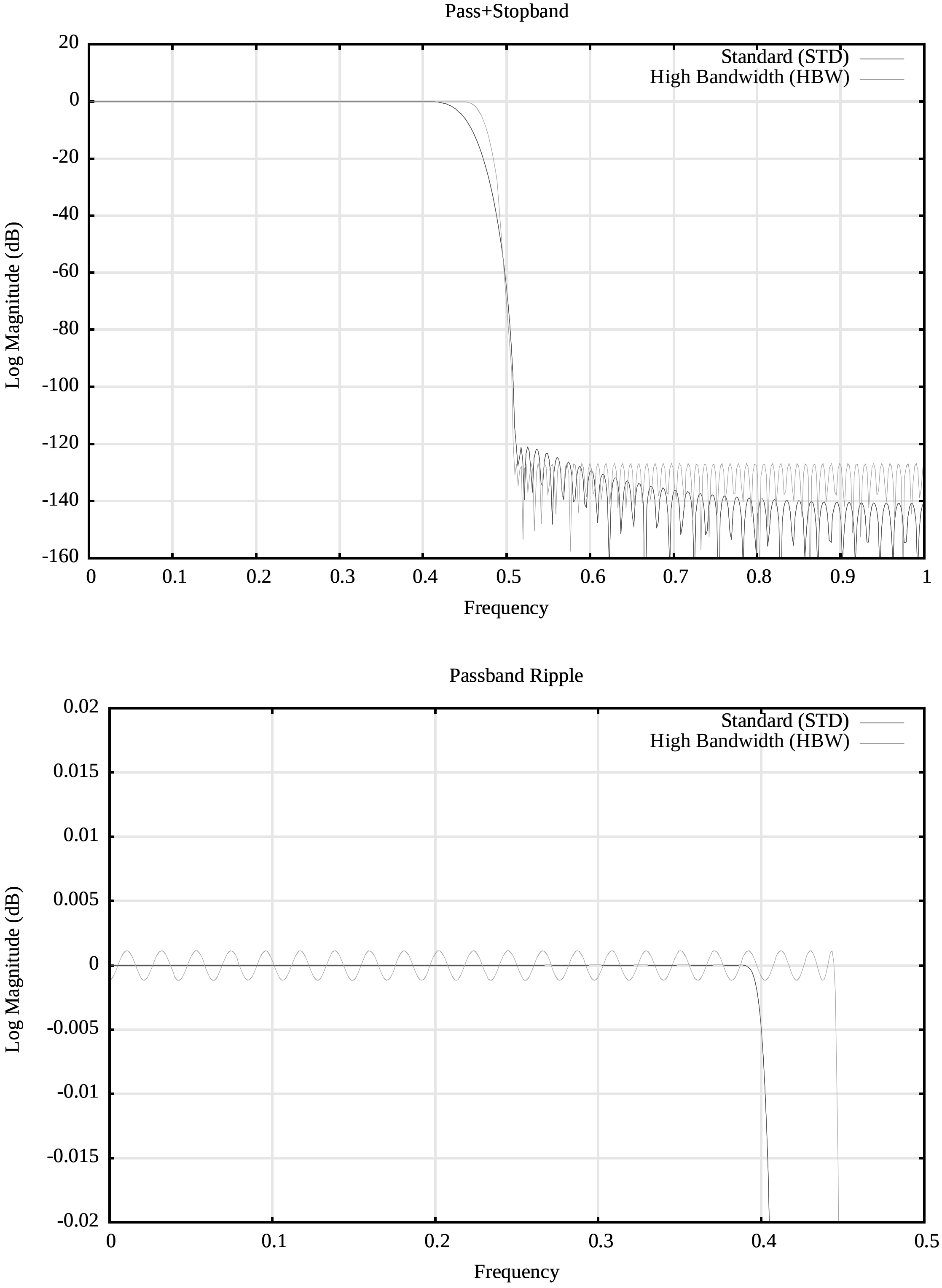


TABLE K-6

Filter Types for Figure K-2

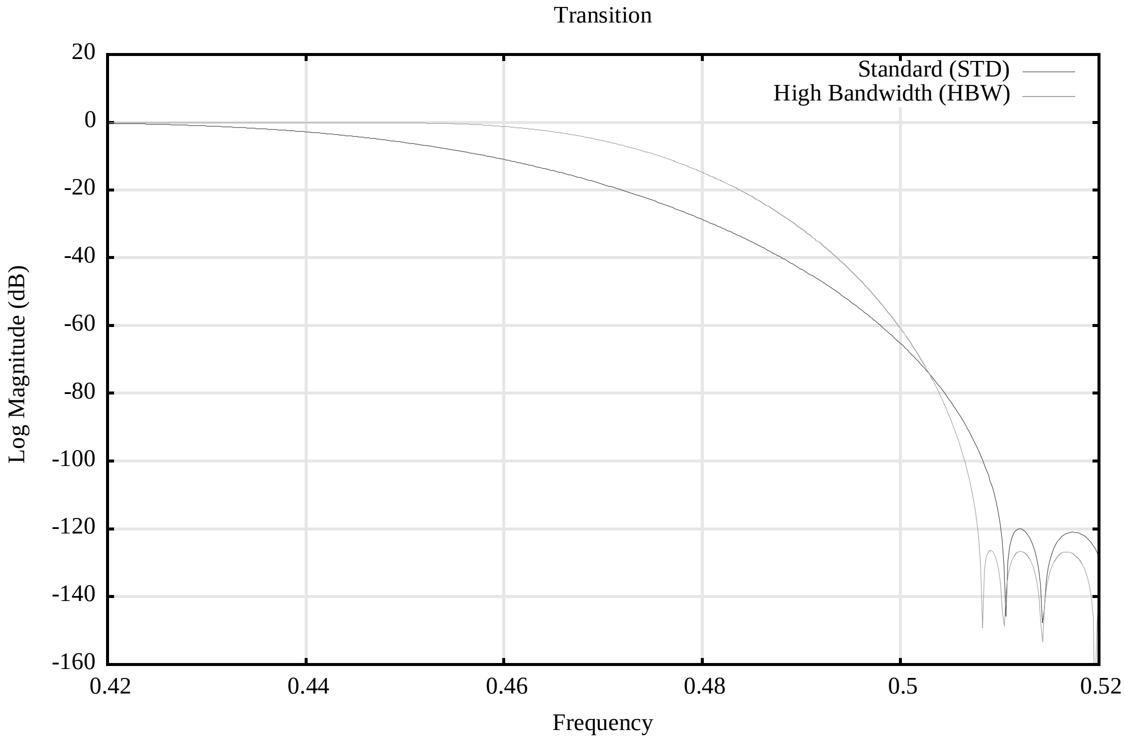
Filter	Type
Standard	24-tap FIR
High Bandwidth	24-tap FIR

FIGURE K-3 **Lowpass Filter Used for 1:2 Upsampling and 1st Stage of 1:4 Upsampling**



Pass+Stopband

Passband Ripple


Transition
Standard (STD)
High Bandwidth (HBW)
Log Magnitude (dB)
20
0
-20
-40
-60
-80
-100
-120
-140
-160
0.42
0.44
0.46
0.48
0.5
0.52
Frequency

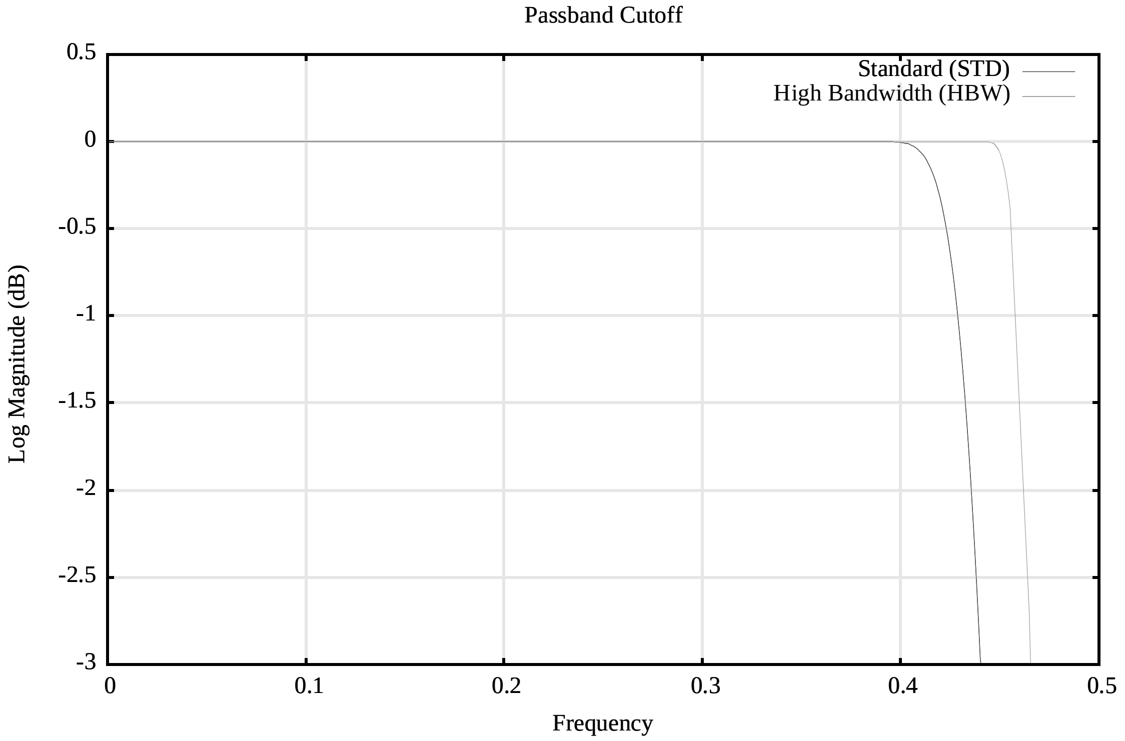

Passband Cutoff
Standard (STD)
High Bandwidth (HBW)
Log Magnitude (dB)
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
0
0.1
0.2
0.3
0.4
0.5
Frequency

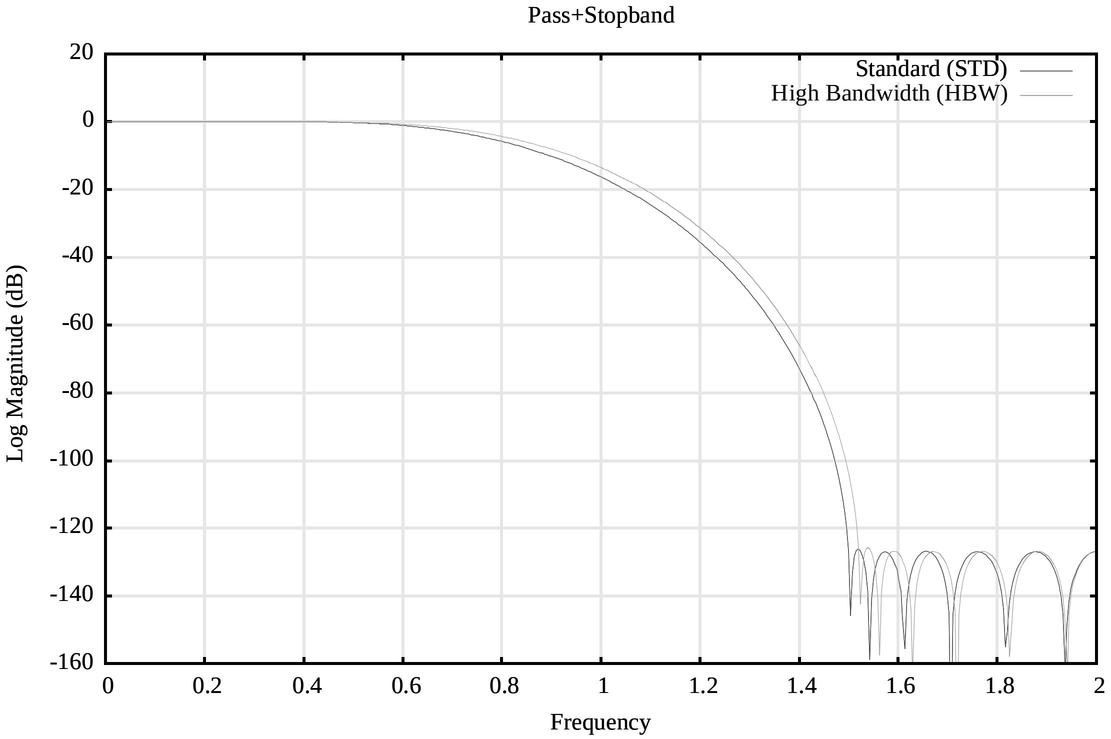
TABLE K-7

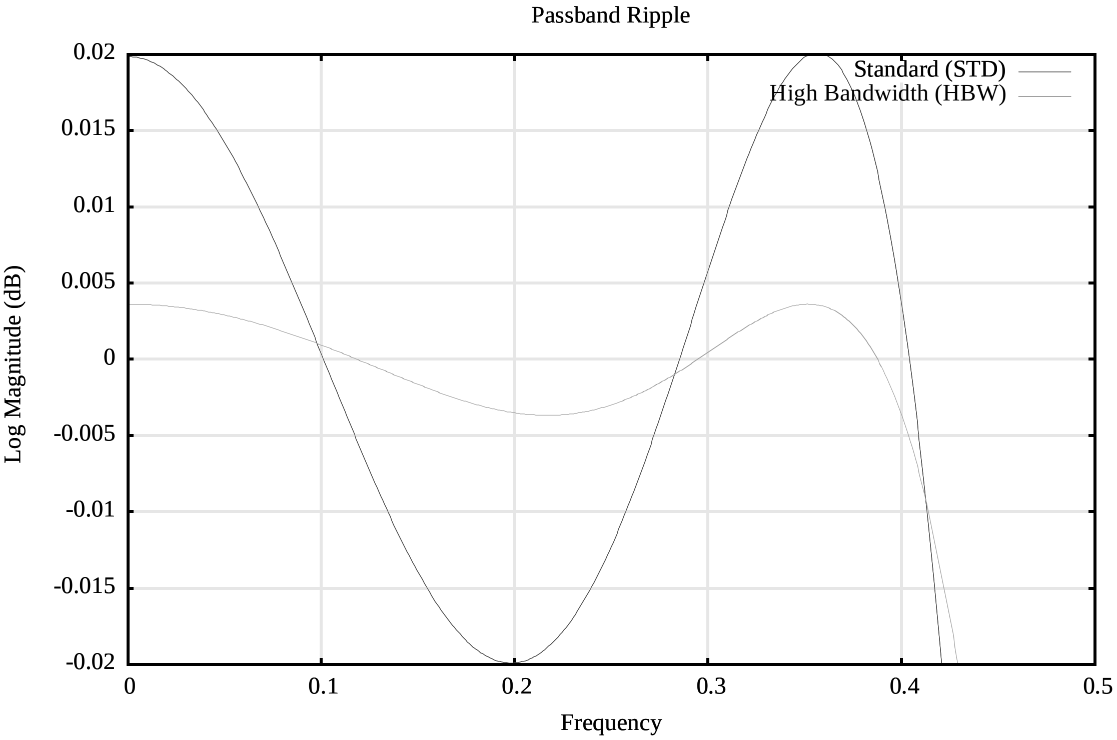
Filter Types for Figure K-3

Filter	Type
Standard	131-tap FIR
High Bandwidth	195-tap FIR

FIGURE K-4

Lowpass Filter Used for 2nd Stage of 1:4 Upsampling




Passband Ripple
Standard (STD)
High Bandwidth (HBW)
Log Magnitude (dB)
0.02
0.015
0.01
0.005
0
-0.005
-0.01
-0.015
-0.02
0
0.1
0.2
0.3
0.4
0.5
Frequency

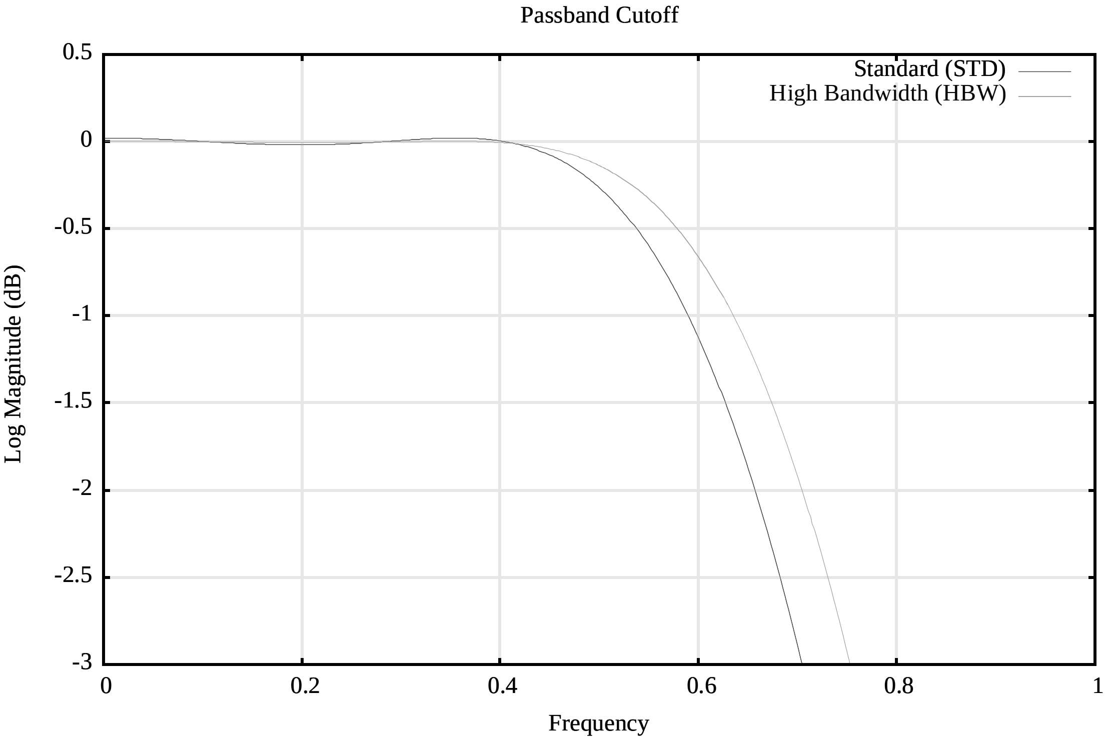

Passband Cutoff
Standard (STD)
High Bandwidth (HBW)
Log Magnitude (dB)
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
0
0.2
0.4
0.6
0.8
1
Frequency

TABLE K-8 **Filter Types for Figure K-4**

Filter	Type
Standard	19-tap FIR
High Bandwidth	19-tap FIR

APPENDIX L Audio Stream Split Number 3

The Audio Stream Split Algorithm is an Audio Stream Processing (ASP) Algorithm with an Application Interface (API) that permits one to exercise control over various aspects of its operation. This appendix explains the details of this component of the Performance Audio Framework (PA/F) API.

An Audio Stream Split Algorithm allows a single audio stream to be split into two with user-controlled mixing during the creation of the secondary audio stream. The Audio Stream Split Algorithm Number 3 (ASS3) is a limited version of the general form. This limited version provides only for the generation of a secondary audio stream having at most Surround4_1 channel configuration.

ASS3 uses the same downmix approach as that used by DM2. Please refer to the DM2 Appendix for specifics of the downmix equations

L.1 Control, Status, and Command Registers

Table L-1 (Audio Stream Split Alpha Code) gives a list of alpha code symbols for the ASS Algorithm. For an explanation of the various forms of register usage, see Section 7.1 (Register Architecture).

Please keep the following in mind when reading this table:

- Default settings for control registers are shaded. Default settings result at power-up or reset.
- Alpha code of type `write` is used to set values in registers, whereas alpha code of type `read` is used to get values from registers.
- Numeric values for the preprocessor symbols shown in this table and appendix are provided in C-language form in the alpha code symbol file `P:\alpha\ass_a.h`.

The ASS Algorithm has no command registers.

TABLE L-1 **Audio Stream Split Alpha Code**

Register	Alpha Code	Description
Status	readASSStatus	Return entire ASS status structure.
Mode	readASSMode	Indicate if Audio Stream Split is enabled or not.
	writeASSModeDisable	Disable Audio Stream Split.
	writeASSModeEnable	Enable Audio Stream Split.
Request	readASSChannelConfiguration-Request	Return ASS Channel Configuration Request Control Register value.
	writeASSChannelConfiguration-Request*X*	Request a channel configuration for the output. See Table 4-3 (Alpha Code for Channel Configuration Registers)
	writeASSChannelConfiguration-RequestStereo	Generate stereo (Lo/Ro) output.
Downmix	readASSChannelConfiguration-Downmix	Return ASS Channel Configuration Downmix Status Register value.
	wroteASSChannelConfiguration-Downmix*X*	A specified channel configuration is generated. See Table 4-3 (Alpha Code for Channel Configuration Registers)
LFE Downmix Volume	readASSLFEDownmixVolume	Return ASS LFE Downmix Volume Control Register value.
	writeASSLFEDownmixVolumeN(NN)	Set ASS LFE Downmix Volume Control Register to NN dB in units of 0.5 dB.
	write ASSLFEDownmixVolumeN(2*10)	Set ASS LFE Downmix Volume Control Register to 10 dB.
LFE Downmix Include	readASSLFEDownmixInclude	Return ASS LFE Downmix Include Control Register value.
	writeASSLFEDownmixIncludeYes	Downmix the LFE Channel into the satellite channels.
	writeASSLFEDownmixIncludeNo	Do not downmix the LFE Channel into the satellite channels.
Center Mix Level	readASSCenterMixLevel	Return ASS Center Mix Level Control Register value.
	writeASSCenterMixLevelN(NN)	Set ASS Center Mix Level Control Register value to NN dB in units of 0.5 dB.
	writeASSCenterMixLevelN(-3*2)	Set ASS Center Mix Level Control Register value to -3 dB.
Surround Mix Level	readASSSurroundMixLevel	Return ASS Surround Mix Level Control Register value.
	writeASSSurroundMixLevelN(NN)	Set ASS Surround Mix Level Control Register value to NN dB in units of 0.5 dB.
	writeASSSurroundMixLevelN(-3*2)	Set ASS Surround Mix Level Control Register value to -3 dB.

L.1.1 Mode

The ASS Mode Control Register controls basic operation of the ASS Algorithm. If zero, operation is disabled, and if non-zero, operation is enabled.

Note: ***Audio Stream Split Disable***

The Audio Stream Split Algorithm ***must not*** *be disabled using the ASS Mode Control Register in PA, as this PA Framework is not (yet) sophisticated enough to handle the resulting signal routing.*

The Audio Stream Split Algorithm may be effectively disabled using the ASS Channel Configuration Request Control Register in PA; the requested channel configuration "None" will produce the desired result.

L.1.2 Channel Configuration

The ASS Channel Configuration Request Control Register is used to request the form that the secondary zone output should take. The ASS Channel Configuration Downmix Status Register reports the form that the output actually does take as a result of the operation of the ASP Algorithm.

The following caveats apply to the current implementation provided by Audio Stream Split Number 3 (ASS3):

- The downmix equations used are as per specified in Appendix Q. Note that "writeASSChannelConfigurationRequestUnknown" is not available, however.
- Do not request a channel configuration that results in more channels than supported by the secondary zone. For example, by default PA17Y has two channels available in the secondary zone, so only
  - None,
  - Phantom0_0 or Stereo
  - Phantom0Stereo_0
  - Phantom0LtRt_0 or StereoLtRt
  - Phantom0Mono_0 or StereoMono

  can be requested. Note that 'Mono' configuration requires that at least five (5) channels be available in the secondary zone, since channels are assigned in the order L, R, Ls, Rs, C, Sw, Lb, Rb.

L.1.3 LFE Downmix Volume

The ASS LFE Downmix Volume Control Register is used to set a gain to be applied to the LFE Channel of ASS input as part of downmix. It is given in units of 0.5 dB, so that the value 2*10 or 0x14 represents 10 dB. The default value of the ASS LFE Downmix Volume Control Register is 10 dB.

One important caveat applies regarding application of the gain specified by the ASS LFE Downmix Volume Control Register:

- This gain is applied only as part of downmix that involves the LFE Channel.

L.1.4 LFE Downmix Include

The ASS LFE Downmix Include Control Register determines whether material from the LFE Channel is downmixed into other channels, as follows:

- If LFEDownmixIncludeYes is set:
  - If ASSChannelConfigurationRequestNone_Y is set, then the LFE input disappears (as is true of other channels), and the secondary stream's channel configuration becomes None_0.
  - If ASSChannelConfigurationRequestMono_Y is set, then the LFE input is first scaled via ASSLFEDownmixVolumeN, and then the scaled LFE is added-into the Mono mix of the satellite channels, forming the Center output. The secondary stream's channel configuration becomes Mono_0.
  - For any other ASSChannelConfigurationRequestXXX_Y, the LFE input is first scaled via ASSLFEDownmixVolumeN, then the scaled LFE is attenuated by an additional 3dB and added to the downmixed L and R output channels. The secondary stream's channel configuration becomes XXXX_0.

  If LFEDownmixIncludeNo is set:
  - If ASSChannelConfigurationRequestXXXX_0 is set, the LFE input is discarded and the secondary stream's channel configuration becomes XXXX_0.
  - if ASSChannelConfigurationRequestXXXX_1 is set, the LFE input is transferred to the secondary zone's Subw output unchanged, and the secondary stream's channel configuration becomes XXXX_1.

For further discussion see Appendix Q.

L.1.5 Center Mix Level

The ASS Center Mix Level Control Register determines the scale applied to the center channel when performing downmix operation. Please refer to Table Q-3 (Downmix Equations) to understand how the center level information is used when performing downmix. The ASS Center Mix Level is indicated by "clev" in the table.

L.1.6 Surround Mix Level

The ASS Surround Mix Level Control Register determines the scale applied to the surround channels when performing downmix operation. Please refer to Table Q-3 (Downmix Equations) to understand how the surround level information is used when performing downmix. The ASS Surround Mix Level is indicated by "slev" in the table.

APPENDIX M PCM Encode Number 2 Algorithm

The PCM Encode Number 2 (PCE2) Algorithm is a Encoder Algorithm with an Application Interface (API) that permits one to exercise control over its operation. This appendix explains the details of this component of the Performance Audio Framework (PA/F) API.

The PCE2 Algorithm provides for floating-point to fixed-point conversion of linear PCM input signals, provides volume control adjustment capabilities, and also performs speaker location delay processing.

The PCE1 Algorithm is . The PCE2 Algorithm is used with -based builds (currently Feature Sets).

M.1 Control, Status, and Command Registers

Table M.1 gives a list of alpha code symbols for the PCE2 Algorithm. For an explanation of the various forms of register usage, see Section 6.1 (Register Architecture).

Please keep the following in mind when reading this table:

- Default settings for control registers are shaded. Default settings result at power-up or reset.
- Alpha code of types `write` and `read` are used to set and get values to and from registers.
- Numeric values for the preprocessor symbols shown in this table and appendix are provided in C-language form in the alpha code symbol file `P:\alpha\pce_a.h`.

The PCE2 Algorithm has no command registers.

TABLE M-1 PCM Encode Number 2 Alpha Code[64]

Register	Alpha Code	Description
Status	readPCEStatus	Return entire PCE status structure.
Control	readPCEControl	Return control registers of PCE status structure.
Mode	readPCEMode	Return PCE Mode Control Register value.
Phase 0 Mode	readPCEModePhase0Mode	Return PCE Mode Phase 0 Mode value.
	writePCEModePhase0ModeDisable	Disable PCE Mode 0.
	writePCEModePhase0ModeEnable	Enable PCE Mode 0.
Volume Mode	readPCEVOLMode	Return PCE Volume Mode (equivalent to readPCEModePhase0Mode).
	writePCEVOLModeDisable	Disable PCE Volume Mode (equivalent to writePCEModePhase0ModeDisable).
	writePCEVOLModeEnable	Enable PCE Volume Mode (equivalent to writePCEModePhase0ModeEnable).
Phase 0 Type	readPCEModePhase0Type	Return PCE Mode Phase 0 type value.
	writePCEModePhase0TypeUnused	Set the Phase 0 type value to unused.
Volume Type	readPCEVOLType	Return PCE Volume Type (equivalent to readPCEModePhase0Type).
	writePCEVOLTypeUnused	Set the PCE Volume Type value to unused (equivalent to writePCEModePhase0TypeUnused).
Phase 1 Mode	readPCEModePhase1Mode	Return PCE Mode Phase 1 Mode value.
	writePCEModePhase1ModeDisable	Disable PCE Mode 1.[a]
	writePCEModePhase1ModeEnable	Enable PCE Mode 1.
Delay Mode	readPCEDELMode	Return PCE Delay Mode (equivalent to readPCEModePhase1Mode).
	writePCEDELModeDisable	Disable PCE Delay Mode (equivalent to writePCEModePhase1ModeDisable).[b]
	writePCEDELModeEnable	Enable PCE Delay Mode (equivalent to writePCEModePhase1ModeEnable).
Phase 1 Type	readPCEModePhase1Type	Return PCE Mode Phase 1 type value.
	writePCEModePhase1TypeUnused	Set the Phase 1 type value to unused.
Delay Type	readPCEDELType	Return PCE Delay Type (equivalent to readPCEModePhase1Type).
	writePCEDELTypeUnused	Set the PCE Delay Type value to unused (equivalent to writePCEModePhase0TypeUnused).
Phase 2 Mode	readPCEModePhase2Mode	Return PCE Mode Phase 2 Mode value.
	writePCEModePhase2ModeDisable	Disable PCE Mode 2.
	writePCEModePhase2ModeEnable	Enable PCE Mode 2.
Output Mode	readPCEOUTMode	Return PCE Output Mode (equivalent to readPCEModePhase2Mode).
	writePCEOUTModeDisable	Disable PCE Output Mode (equivalent to writePCEModePhase2ModeDisable).
	writePCEOUTModeEnable	Enable PCE Output Mode (equivalent to writePCEModePhase2ModeEnable).

Register	Alpha Code	Description
Phase 2 Type	readPCEModePhase2Type	Return PCE Mode Phase 2 type value.
	writePCEModePhase2TypeUnused	Set the Phase 2 type value to unused.
Output Type	readPCEOUTType	Return PCE Output Type (equivalent to readPCEModePhase2Type).
	writePCEOUTTypeUnused	Set the PCE Output Type value to unused (equivalent to writePCEModePhase2TypeUnused).
Output Exception Detection	readPCEExceptionDetectMode	Return PCE Exception Detection Mode ie. Queries functionality of floating point exception detection (at fxed point conversion stage of audio sample buffer, eg. QNAN,+Inf,-Inf,NaN,SnaNetc.) is enabled or disabled.
	writePCEExceptionDetectDisable	Disarm floating point exception detection functionality.
	writePCEExceptionDetectEnable	Arm floating point exception detection functionality.
	readPCEExceptionDetectFlag	Signal detection of floating point exceptions in audio sample buffer.
	writePCEExceptionDetectFlagOff	Flag no floating point exceptions in audio sample buffer.
	writePCEExceptionDetectFlagOn	Flag detction of floating point exceptions in audio sample buffer. This flag is sticky ; ie one need to set explitly make it Off for a second use.
	readPCEExceptionDetectMute	Return mute functionality on exception detection.
	writePCEExceptionDetectUnmute	Disarm muting on detection of exception.
	writePCEExceptionDetectMute	Arm muting on detection of exception.
Output Clip Detection	readPCEClipDetectFlag	Return Clip detection flag. Flag will be ON if the sample value is higher than +6dB in floating point domain.
	writePCEClipDetectFlagOff	Flags sample value is NOT higher than +6dB in floating point domain.
	writePCEClipDetectFlagOn	Flags sample value IS higher than +6dB in floating point domain. This flag is sticky ; ie one need to set explitly make it Off for a second use
Phase 3 Mode	readPCEModePhase3Mode	Return PCE Mode Phase 3 Mode value.
	writePCEModePhase3ModeDisable	Disable PCE Mode 3.
	writePCEModePhase3ModeEnable	Enable PCE Mode 3.
Phase 3 Type	readPCEModePhase3Type	Return PCE Mode Phase 3 type value.
	writePCEModePhase3TypeUnused	Set the Phase 3 type value to unused.
Phase 4 Mode	readPCEModePhase4Mode	Return PCE Mode Phase 4 Mode value.
	writePCEModePhase4ModeDisable	Disable PCE Mode 4.
	writePCEModePhase4ModeEnable	Enable PCE Mode 4.
Phase 4 Type	readPCEModePhase4Type	Return PCE Mode Phase 4 type value.
	writePCEModePhase4TypeUnused	Set the Phase 4 type value to unused.

[64] The PCE2 Algorithm Speaker Location Delay (PCEDEL) alpha commands are located in Appendix I (Speaker Location Delay).

Register	Alpha Code	Description
Phase 5 Mode	readPCEModePhase5Mode	Return PCE Mode Phase 5 Mode value.
	writePCEModePhase5ModeDisable	Disable PCE Mode 5.
	writePCEModePhase5ModeEnable	Enable PCE Mode 5.
Phase 5 Type	readPCEModePhase5Type	Return PCE Mode Phase 5 type value.
	writePCEModePhase5TypeUnused	Set the Phase 5 type value to unused.

[a] Once disabled, the Phase 1 Mode (Delay Mode) cannot be re-enabled.

[b] Once disabled, the Phase 1 Mode (Delay Mode) cannot be re-enabled.

M.1.1 Mode

The PCE Mode Control Register controls basic operation of the PCE Algorithm. This register is only a status register. The PCE Mode cannot be disabled.

M.1.2 Phase 0 Mode

The Phase 0 Mode of the PCE Algorithm is defined as the Volume Phase. Volume Control processing within the system occurs at this first stage, or phase, of PCM Encode processing. For more information on the Volume Control application interface, please see Section 5.4 (Volume Control). Note that if the PCE Phase 0 Mode is disabled, undesired behavior will occur.

M.1.3 Phase 1 Mode

The Phase 1 Mode of the PCE Algorithm is defined as the Delay Phase. Speaker Location Delay processing within the system occurs at the second stage, or phase, of PCM Encode processing.

Even though the alpha command writePCEDELModeEnable (writePCEModePhase1ModeEnable) is defined, it must ***not*** be used to re-enable speaker location delay following writePCEDELModeDisable (writePCEModePhase1ModeDisable). Under some situations, re-enabling delay via writePCEDELModeEnable (writePCEModePhase1ModeEnable) may appear to work, but under others it is known to induce a system crash. The only supported way to disable delays so that they may be subsequently re-enabled is to set the individual speaker delays to zero; subsequently, speaker location delay can be introduced by setting the individual speaker delays to the desired values.

More information on the Speaker Location Delay interface using the PCE2 Algorithm is located in Appendix I (Speaker Location Delay).

M.1.4 Phase 2 Mode

The Phase 2 Mode of the PCE Algorithm is defined as the Output Phase. In the Output Phase the audio samples for all of the supported output channels are converted from floating-point representation to fixed-point for interface with external peripherals such as digital-to-analog (D/A) converters. Note that if the PCE Phase 2 Mode is disabled, undesired behavior will occur.

M.1.5 Phase 3 Mode

The Phase 3 Mode of the PCE Algorithm is currently unused.

M.1.6 Phase 4 Mode

The Phase 4 Mode of the PCE Algorithm is currently unused.

M.1.7 Phase 5 Mode

The Phase 5 Mode of the PCE Algorithm is currently unused.

M.1.8 Phase 0-5 Type

The Phase 0 through Phase 5 Type registers of the PCE Algorithm are currently unused.

M.1.9 Encoder Bypass

The PCE2 algorithm supports Bypass functionality. This functionality allows for the data to be passed “as is” without any modification. The Bypass functionality can be enabled by using the alpha code writeENCBypassEnable. The Bypass functionality can be disabled by using the alpha code writeENCBypassDisable. By default, the Bypass functionality is disabled.

The Bypass functionality is considered useful in circumstances when the input data to PCE2 is not floating point, and thus it is required that the PCE2 algorithm doesn’t operate on it. If the Bypass functionality is enabled, some of the other functionalities of PCE2 are disabled. Specifically, Volume Control (including Volume Ramp) is disabled.

For the PCE2 Bypass functionality to work correctly, it is required that the PCE Delay mode be disabled. Please note the PCE Delay (Phase 1) is not automatically disabled if Bypass feature is enabled. The PCE Delay mode needs to be explicitly disabled by using the alpha code writePCEModePhase1ModeDisable (or, identically, writePCEDELModeDisable).

PCE2 algorithm is designed to reduce the system memory requirement by converting floating point input data to 24bit fixed point data. The 24bit fixed point data is used by the PCE2 Delay stage and the delayed 24bit data is provided to the output audio driver. This reduces the memory required for the delay line and also reduces the memory required for storing the output samples. This feature adds extra limitation when using the Bypass functionality. If the Bypass functionality is desired, it is rquired not just to disable the PCE2 Delay mode, but also increase the memory allocation for storing the output samples and configure the output audio driver to work with 32bit data. For more information on how these configurations can be done, please refer the FAQ.pdf document provided as part of the release.

PCE2 Bypass functionality doesn’t affect the ENCChannelMap features. Specifically if the ENCChannelMapFrom for a particular channel is set to -1, the PCE2 algorithm will provide Zero-valued data for the channel. Similarly, if the ENCChannelMapFrom for a particular channel is set to -2, no audio data is provided for the channel.

APPENDIX N System specific

14.0.1 DTS Lock Feature

As per the guidelines from DTS Inc. regarding using DTS technologies, PAF implements feature to lock and unlock usage of them. If any of the DTS technologies are used in the system, then it requires a unique key to unlock and use them. Without this key, the system will not be able to operate properly. The SDK gets distributed without any key and so SDK users need to obtain a key before they can use DTS technologies. However, a system need not have this key if it doesn't use any of the DTS technologies.

14.0.2 System CPU Load

The system stream measures the 'instantaneous' CPU load value over intervals of approximately one half-second. From these successive 'instantaneous' loads, a 'peak' load is accumulated; the 'peak' represents the maximum 'instantaneous' load since the system was restarted or the peak was most-recently cleared. Alpha commands are provided, as shown in Table 14-1 (Alpha Code for System CPU Load), to read the 'instantaneous' and 'peak' loads, and to clear the 'peak' load reading.

The load readings provided by the system stream and the load readings provided by the CCS load graph may differ slightly, but the values returned by the system stream are more accurate.

When the CPU is loaded more than 100% (so that real time operation is broken), the values returned by following alpha codes will be incorrect, due to load-shedding.

TABLE 14-1 **Alpha Code for System CPU Load**

Register	Alpha Code	Description
CPU load	readSYSCpuLoad	Return the integral part of the ‘instantaneous’ CPU load in 8-bit Q0 format. For a CPU load of 23.56%, returns 0x17 (23).
	readSYSCpuLoadQ8	Return the ‘instantaneous’ CPU load in 16-bit Q8 format, rounded to integral percentage. For a CPU load of 23.00%, returns 0x1700.
Peak CPU load	readSYSPeakCpuLoad	Return the integral part of the ‘peak’ CPU load in 8-bit Q0 format. For a peak CPU load of 23.56%, returns 0x17 (23).
	readSYSPeakCpuLoadQ8	Return the ‘peak’ CPU load in 16-bit Q8 format, rounded to integral percentage. For a peak CPU load of 23.00%, returns 0x1700.
	writeSYSPeakCpuLoadClear	Set the ‘peak’ CPU load to zero, restarting peak load accumulation. The peak load reading of 0 will be overwritten when the system stream next measures the ‘instantaneous’ load. Successive ‘instantaneous’ load readings will replace the ‘peak’ load if larger.

14.0.3 Memory Usage Information

Alpha code for use with the control and status registers described in this section is given in Table 14-2 (Alpha Code for Memory Usage Information).

These alpha commands return the requested memory information, in units of bytes, as a 32-bit result, in the form “LSW, MSW”, where LSW is the 16 least-significant bits of the result, and MSW is the 16 most-significant bits.

Note: ***readACPStat...***

Unlike most “read” alpha codes, which return a corresponding “write” alpha code, “readACPStat...” returns two words of binary data, which is not *alpha code.*

For further information, see section “4.4 Alpha Code Type 0 Write (Function Invocation)” in the Performance Audio Messaging Application Protocol Application Report.

TABLE 14-2 **Alpha Code for Memory Usage Information**

Register Set	Alpha Code	Description
IRAM Info.	readACPStatIRAMSize	Heap size
	readACPStatIRAMUsed	Memory used
	readACPStatIRAMLength	Length of largest contiguous block
	readACPStatIRAMFree	Memory free (=size - used)

Register Set	Alpha Code	Description
SDRAM Info.[a]	readACPStatSDRAMSize	Heap size
	readACPStatSDRAMUsed	Memory used
	readACPStatSDRAMLength	Length of largest contiguous block
	readACPStatSDRAMFree	Memory free (=size - used)
L3RAM[b]	readACPStatL3RAMSize	Heap size
	readACPStatL3RAMUsed	Memory used
	readACPStatL3RAMLength	Length of largest contiguous block
	readACPStatL3RAMFree	Memory free (=size - used)

[a] In systems with no SDRAM, these alpha codes will instead return information about IRAM memory usage.

[b] In systems with no L3RAM heap, these alpha codes will instead return information about IRAM memory usage.

14.0.4 Build Identification

Alpha code for use with the control and status registers described in this section is given in Table 14-3 (Alpha Code for Build Identification Information).

The identificaiton alpha command returns the requested build identification information in hexadecimal. The information is provided in three subsets:

- Topology (I = 1; 2 = D; 3 = Y; 4 = H; 5 = Z)
- Feature Set (7 or 8)
- Date (YYMMDD)

For example, if the value is returned is:

```
0xcc04, 0x0004, 0x0815, 0x1705
```

The last two alpha words should be reversed and interpreted as:

- Topology: I (“1”)
- Feature Set: 7 (“7”)
- Date: August 15, 2005 (“050815”)

The first 16-bit word contains the MM and the DD. The second 16-bit word contains the Topology, Feature Set, and Year.

TABLE 14-3 **Alpha Code for Build Identification Information**

Register Set	Alpha Code	Description
Identification	readIDIdentification	Build Identification (In hexadecimal: topology (1=I, 2=D, 3=Y, 4=H, 5=Z), Feature Set, Date (YYMMDD)

APPENDIX O EVM Specific

As mentioned in Chapter 1, Introduction the PA software is supposed to run on a hardware evaluation module (EVM) provided by Spectrum Digital. More information regarding the layout and schematics of the various hardware releases are available on the Spectrum Digital website. What follows in this chapter is an explaination of aspects of the PA software specific to the EVM.

O.0.1 IOS shortcuts

As explained in Table 3.1.1 (Audio Input Shortcuts) and Table 4.2.3 (Audio Output IOS) the input and output shotcuts or IOS are a set of alpha codes which prepares PA for a given input and output. In this section we will examine one input and one outout IOS in detail.

O.0.1.1 Input IOS

Table O-1 (execPAYInDigital) explains the various alpha codes that are part of the IOS shortcut.

TABLE O-1 **execPAYInDigital**

Alpha code
`writeDECSourceSelect-None`
`writePA3Await(rb32DECSourceDe-code,ob32DECSourceDecodeNone)`

Alpha code
writeIBUnknownTimeoutN(2*2048)
writeIBScanAtHighSampleRateModeDisable
writePCMChannelConfigurationProgramStereoUnknown
writePCMScaleVolumeN(0)
writeDECChannelMapFrom16(0,1,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3)
writeIBEmphasisOverrideDisable
writeIBPrecisionDefaultOriginal
writeIBPrecisionOverrideDetect
writeIBSampleRateOverrideStandard
writeIBSioSelectN(DEVINP_DIR)
writeDTSHDSpeakerRemapModeDisable
wroteDECSourceProgramUnknown
writeDECSourceSelectAuto
0xcdf0
execPAYInDigital

O.0.1.2 Output IOS

Table O-2 (execPAIOutAnalog) explains the various alpha codes that are part of the IOS shortcut.

TABLE O-2 **execPAIOutAnalog**

Alpha code
rb32DECSourceSelect_3
writeDECSourceSelectNone

Alpha code
writePA3Await(rb32DECSourceDecode,ob32DECSourceDecodeNone)
writeOBSioSelectN(1)
writeENCChannelMapTo16(3,7,2,6,1,5,0,4,-3,-3,-3,-3,-3,-3,-3,-3)
wb32DECSourceSelect_3
0xcdf0
execPAIOutAnalog

O.0.1.3 Special Alpha Codes used by Input/Output Switching (IOS)

There are special alpha codes used when selecting the input and output configurations as part of *Input / Output Switching (IOS)*.

See Table 15-3 (Special Alpha Codes used by Input/Output Switching) for a list of these special alpha codes.

TABLE 15-3 **Special Alpha Codes used by Input/Output Switching**

Alpha Code	Hex Representation	Description
rb32OBSioSelect	0xc022,0x0581	Store the OBSioSelect register.
rb32DECSourceDecode	0xc024,0x0b81	Store the DECSourceDecode register.
rb32DECSourceSelect_3	0xc024,0x09b1	Store the DECSourceSelect register.
wb32DECSourceSelect_3	0xc024,0x09f1	Restore the DECSourceSelect register.
writePA3Await(RB32,WB32)	0xcd0b,5+2,0x0204,200,1,WB32,RB32	Used to await a given system state.
writeNoResponseN(N)	0xcd0b,5+N,0x0004,0,0,0,0	Used as *prefix* to alpha code, to discard response from that alpha code.
stream1	0xcd09,0x0400	Select First Stream.
stream2	0xcd09,0x0401	Select Second Stream.
stream3	0xcd09,0x0402	Select Third Stream.

Additional special alpha codes (which are not currently used in Input/Output Switching (IOS)) are described in Table 15-4 (Other Special Alpha Codes).

TABLE 15-4 **Other Special Alpha Codes**

Alpha Code	Hex Representation	Description
rb32DECChannelConfigurationProgram	0xc024,0x1444	Store the DECChannelConfigurationProgram register.
rb32DECSampleRate	0xc024,0x0849	Store the DECSampleRate register.
rb32DECSourceProgram	0xc024,0x0a49	Store the DECSourceProgram register.
rb32ENCChannelConfigurationStream	0xc025,0x0649	Store the ENCChannelConfigurationStream register.
rb32ENCSampleRate	0xc025,0x0c44	Store the ENCSampleRate register.

15.0.2 Purpose of Special IOS Alpha Codes

These alpha codes are used in input and output signal selection.

When performing I/O selection, it is required that the framework doesn't make any data request to the input driver or transfer any data to the output driver. This is needed so that the input or output driver configuration can be changed without any undesired effects. To realize this, the writeDECSourceSelectNone alpha code is issued. This alpha code informs the framework that no decoding is intended. The framework in response to this command stops all audio processing operations (decoding, autodetection, etc). The framework makes the Source Decoder NONE when it has stopped all the audio processing operations. Thus, reading back the source decode and verifying that it has become NONE (using the alpha code readDECSourceDecode) guarantees that the I/O configuration can be changed without any undesired affect.

For the Input configuration selection, the input switching alpha codes have the DECSourceSelect alpha codes set by default. For example, Digital Input selection includes writeDECSourceSelectAuto whereas Analog Input selection includes writeDECSourceSelectPCM.

For Output selection the DECSourceSelect is not required to be explicitly set, yet the DECSourceSelect register must be restored to what it was before. Therefore, the rb32DECSourceSelect_3 and wb32DECSourceSelect_3 alpha codes are required to save and restore the DECSourceSelect register value while the output configuration is being changed.

15.0.3 Construction of Special IOS Alpha Codes

15.0.3.1 rb32DECSourceSelect_3 and wb32DECSourceSelect_3 Definition

The first short of the alpha code is as follows:

0xc024 -> 1100 0000 0010 0100

TABLE 15-5 **Description of rb32DECSourceSelect_3/wb32DECSourceSelect_3 alpha codes**

Word	Bits	Name	Description
0	15 - 14	Legacy	Equal to 11_b[a] to indicate non-legacy code.
	13 - 12	Series	Set to 00 for standard system alpha codes. Set to 11 for customer alpha codes.
	11	Read/Write	0 to indicate Read.
	10 - 8	Type	0 means Type 0 alpha code
	7 - 0	Beta Number	Beta number is 0x24 (DEC global structure (pafdec_a.h)
1	15 - 8	Offset	0x09 means this is the DECSourceSelect field
	7 - 0	Selection	0xb1 (rb32DECSourceSelect_3), 0xf1 (wb32DECSourceSelect_3)

[a] The subscript **b** is used to denote binary, or base 2, numbers.

For Type 0 read alpha codes, data is read from the Beta Unit and the 8-bit offset. (In the case of rb32DECSourceSelect_3, the Beta Unit is 0x24 and the Offset is 0x09).

The results of the read operation are handled differently depending upon the lower 8 bits (In this case 0xb1 or 0xf1).

Below is a description of how the lower 8 bits are handled:

TABLE 15-6 **Description of Explicit Read field for Type 0 Alpha Code**

Bit 7	Bit 6	Bits 5-4	Bits 3-0	Description
1	0	*N*	0x0	32-bit argument N set from address.
1	0	*N*	0x1	32-bit argument N set from 8-bit unsigned value.
1	1	*N*	0x1	8-bit location set from argument N.
1	0	*N*	0x2	32-bit argument N set from 16-bit unsigned value.
1	1	*N*	0x2	16-bit location set from argument N.
1	0	*N*	0x4	32-bit argument N set from 32-bit value.
1	1	*N*	0x4	32-bit location set from argument N.
1	0	*N*	0x9	32-bit argument N set from 8-bit signed value.
1	0	*N*	0xa	32-bit argument N set from 16-bit signed value.

For the alpha code rb32DECSourceSelect_3 lower byte is : 0xb1. Which means:

32-bit argument N set from 8-bit unsigned value.

For the alpha code ob32DECSourceSelect_3 lower byte is : 0xf1. Which means:

8-bit location set from argument N.

The Alpha File Processing (AFP) task has a few variables called Argument variables. These variables are used to hold some values. The alpha code rb32DECSourceSelect_3 sets the argument 3 register with the current DECSourceSelect value. The alpha code wb32DECSourceSelect_3 sets the DECSourceSelect value with the argument 3.

In other words, the above alpha codes work to save the DECSourceSelect value and restore it. The alpha codes are designed for basically pushing the DECSourceSelect value and popping it back once the output configuration is set.

15.0.3.2 rb32DECSourceDecode Definition

#define rb32DECSourceDecode 0xc024,0x0b81

The value “0xc0” signifies Alpha Code Type 0 Read (explicit binary read, which is defined in Section 15.0.3.1).

The value “0x24”signifies the PA/F Decoder Beta ID.

The value “0x0b” represents the offset pointing to the DECSourceDecode varaible.

Argument 1 is set with 32 bit value returned from Beta ID 0x24 and offset 0x0b.

The value read is the DECSourceDecode variable.

15.0.3.3 wb32DECSourceDecode Definition

TBD.

15.0.3.4 rb32OBSioSelect Definition

TBD.

15.0.3.5 stream1, stream2, and stream3 Definition

#define stream1 0xcd09,0x0400 /* First stream */

#define stream2 0xcd09,0x0401 /* Second stream */

#define stream3 0xcd09,0x0402 /* Third stream */

The value “0xcd09” signifies Alpha Code Type 5, subtype 9 which is described as:

“Alpha code type 5-9 is used to set the registers associated with an alpha code processing unit.”

This alpha code sets or clears the Beta Prime Value Control register.

This is used so that identical alpha codes can be used to access the memory-mapped control, status, and command registers in multiple audio streams.

15.0.3.6 writePA3Await Description

#define writePA3Await(RB32,WB32) 0xcd0b,5+2,0x0204,200,1,WB32,RB32

The value “0xcd0b” signifies and Alpha Code Type 5-11 write which is described as:

"Alpha code type 5-11 is used to await a given system state in order to synchronize processing unit."

5+2 -> Lambda field indicates the length of alpha code type 5-11 operation

0x02 -> the Xi field indicates the comparison operation. In this case the value “2” implies '==', or equals.

0x04 -> the Phi field indicates the function implemented to create the delay operation. In this case 'sleep'

200 -> the Kappa field gives the repeat count

1 -> the Tau field gives the time to delay in milliseconds

The encapsulated alpha code values (RB32, WB32) are compared with one another. If the result of the comparison is true, the wait is over (in this case, if RB32 = WB32). If the result is false, then the system waits a given time interval (defined by Tau field) and tries again. If after a given number of times (defined by the Kappa field) the result is still false, the system returns an error.

If there is no encapsulated alpha code (in other words if there was a NULL argument to the writePA3Await function), the alpha code will simply implement a delay of Kappa * Tau in length and return.

15.0.3.7 writeNoResponseN Description

```
#define writeNoResponseN(N)0xcd0b,5+N,0x0004,0,0,0,0
```

This alpha code is used to “prefix” other alpha code, of length N, so as to prevent any response being returned from that alpha code. This is useful, in particular, when *invoking* IOS shortcuts (but not useful in *defining* IOS shortcuts).

This is an “Alpha code type 5-11”, as discussed in Section 15.0.3.6, comprising the following fields:

5+N -> the Lambda field indicates the length, in 16-bit words, of alpha code type 5-11 “payload” (i.e., items following this field), where N is the length of the alpha code to be executed, e.g., an IOS shortcut

0x00 -> the (8-bit) Xi field indicates the comparison operation. In this case the value “0” implies “always true”.

0x04 -> the (8-bit) Phi field indicates the function implemented to create the delay operation. In this case 'sleep' (though this isn’t really meaningful, given Kappa = 0)

0 -> the (16-bit) Kappa field specifies the repeat count (only need to try once, since “always true”)

0 -> the (16-bit) Tau field specifies the time to delay, in milliseconds (actually, this value is arbitrary, since not a repeated operation)

0,0 -> the (32-bit) Delta field specifies the “compare-to value” (this, too, is arbitrary, since comparison is “always true”)

As an example, compare the following cases, with & without `writeNoResponseN()`.

Normal IOS invocation, which gives 1-word response after operation completed:

```
P:\ alpha ... execPA17IInDigital0
alpha execPA17IInDigital0
```

Alternative IOS invocation to avoid any response:

```
P:\> alpha ... writeNoResponseN(1),execPA17IInDigital0
alpha /* none */
```

The argument “1” is used above because that’s the length, in 16-bit words, of alpha-code shortcut (invocation) `execPA17IInDigital0`.

1. SelectNone has successfully taken effect. This return value (0x0001,0x0000) is the data portion of the alpha code writeDECChannelConfigurationRequestNone.

   writePA3Await(rb32DECSourceDecode,ob32DECSourceDecodeNone)

2. Perform all actions to select the new output.

   writeOBSioSelectN(DEVOUT_DAC)

3. Restore decoding of input to the state saved in step 1 above.

   wb32DECSourceSelect_3