TA Document 1999008
AV/C Audio Subunit Specification 1.0

October 24, 2000

Sponsored by:
1394 Trade Association

Accepted for Release by:
1394 Trade Association Board of Directors.

Abstract:
This document specifies data structures and a command set to be used to control consumer electronic audio devices and subunits. The command set builds upon an extensive body of standards work, underway and completed. It also concerns itself with the syntax and semantics of commands transmitted by controllers to AV-audio subunits and the resultant actions that occur at the audio subunit. Additionally the data structures revealing the logical connectivity within the audio subunit are specified. This document also describes data structures that indicate which audio data formats can be processed by audio subunits.

The aim is to accommodate highly complex as well as simple devices.

Keywords:
Audio, Video, 1394, Digital, Interface, Audio Control.

1394 Trade Association Specifications are developed within Working Groups of the 1394 Trade Association, a non-profit industry association devoted to the promotion of and growth of the market for IEEE 1394-compliant products. Participants in working groups serve voluntarily and without compensation from the Trade Association. Most participants represent member organizations of the 1394 Trade Association. The specifications developed within the working groups represent a consensus of the expertise represented by the participants.

Use of a 1394 Trade Association Specification is wholly voluntary. The existence of a 1394 Trade Association Specification is not meant to imply that there are not other ways to produce, test, measure, purchase, market or provide other goods and services related to the scope of the 1394 Trade Association Specification. Furthermore, the viewpoint expressed at the time a specification is accepted and issued is subject to change brought about through developments in the state of the art and comments received from users of the specification. Users are cautioned to check to determine that they have the latest revision of any 1394 Trade Association Specification.

Comments for revision of 1394 Trade Association Specifications are welcome from any interested party, regardless of membership affiliation with the 1394 Trade Association. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally, questions may arise about the meaning of specifications in relationship to specific applications. When the need for interpretations is brought to the attention of the 1394 Trade Association, the Association will initiate action to prepare appropriate responses.

Comments on specifications and requests for interpretations should be addressed to:

Editor, 1394 Trade Association
Regency Plaza Suite 350
2350 Mission College Blvd.
Santa Clara, Calif. 95054, USA

Table of Contents

List of Figures

List of Tables

This page is left intentionally blank

1. Overview

1.1 Purpose and Scope

This document specifies data structures and a command set to be used to control consumer electronic audio devices and subunits. The command set builds upon an extensive body of standards work, underway and completed. Primitive function block types are defined to accommodate a wide range of possible audio topologies.

This specification concerns itself with the syntax and semantics of commands transmitted by controllers to AV-audio subunits and the resultant actions that occur at the audio subunit. Additionally the data structures revealing the logical connectivity within the audio subunit are specified. This document also describes data structures that indicate which audio data formats can be processed by audio subunits.

2. References

The following standards contain provisions, which through reference in this document, constitute provisions of this standard. All the standards listed are normative references. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

[R1] ISO/IEC 13213-1994 (ANSI/IEEE1212-1994): Information Technology-Microprocessor Systems-Control and Status Register (CSR) architecture for Microcomputer Buses.

[R2] IEEE1394-1995 Standard for a High Performance Serial Bus

[R3] IEC 61883 part 1-5 Consumer audio /video equipment - Digital interface

[R4] AV/C Digital Interface Command Set General Specification, version 3.0, April 1998

[R5] IEC-PAS 61883-6, Consumer audio/video equipment – Digital interface – part 6: Audio and music data transmission protocol

[R6] ANSI/IEEE754 Floating Point Format

[R7] ITU G.711

[R8] IEC 61937, Digital audio - Interface for non-linear PCM encoded audio bitstreams applying IEC60958

[R9] IEC 60958 Digital Audio Interface

[R10] ISO/IEC 13818-3:1995, Information Technology -- Generic Coding of Moving Pictures and Audio - Part 3: Audio

[R11] ISO/IEC 13818-1:1995, Information Technology -- Generic Coding of Moving Pictures and Audio - Part 1: Systems

[R12] ISO/IEC 11172-1, Information technology -- Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s -- Part 3: Audio

[R13] ISO11639, “Three-letter code for the representation of names of languages for information interchange”, developed by a Joint Working group of ISO TC37/SC2 and TC46/SC2 (see also ISO/IEC JTC1 N1333).

[R14] ISO10646:1993, “Information Technology – Universal Multiple-Octet Coded Character Set (UCS).”

[R15] UNICODE, The Unicode Standard: Version 2.0, The Unicode Consortium, Addison-Wesley Developers Press, 1996.

[R16] DTS Digital Surround, Licensee Manual, July 1998, v1.0

[R17] DTS Coherent Acoustics Decoder, March 30 1998, v1.0

[R18] Digital Audio Compression Standard (AC-3), The Advanced Television Systems Committee, doc A/52, Dec 20 ‘95

[R19] “Multi-channel Digital Audio Decoding System for Consumer Products” Licensee Information Manual Version 2.0 April, 1997

3. Definitions

3.1 Conformance Levels

3.1.1 expected: A key word used to describe the behavior of the hardware or software in the design models *assumed* by this Specification. Other hardware and software design models may also be implemented.

3.1.2 may: A key word that indicates flexibility of choice with *no implied preference*.

3.1.3 shall: A key word indicating a mandatory requirement. Designers are *required* to implement all such mandatory requirements.

3.1.4 should: A key word indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase *is recommended*.

3.1.5 reserved fields: A set of bits within a data structure that are defined in this specification as reserved, and are not otherwise used. Implementations of this specification shall zero these fields. Future revisions of this specification, however, may define their usage.

3.1.6 reserved values: A set of values for a field that are defined in this specification as reserved, and are not otherwise used. Implementations of this specification shall not generate these values for the field. Future revisions of this specification, however, may define their usage.

NOTE —The IEEE is investigating whether the “may, shall, should” and possibly “expected” terms will be formally defined by IEEE. If and when this occurs, draft editors should obtain their conformance definitions from the latest IEEE style document.

3.2 Glossary of Terms

3.2.1 (Audio) Channel: An audio channel is a serial stream of audio data that when presented serially in its entirety on a single speaker in human intelligible form reproduces a self contained and consistent signal.

3.2.2 (Audio) Channel Cluster: Group of logical audio channels that carry tightly related synchronous audio information. A stereo audio stream is a typical example of a two-channel audio channel cluster.

3.2.3 Audio Control Attribute: Parameter of an Audio Control. Examples are Current, Minimum, Maximum and Resolution attributes of a Volume Control.

3.2.4 Audio Control: Logical object that is used to manipulate a specific audio property. Examples are Volume Control, Mute Control, etc.

3.2.5 Audio data stream: Transport medium that can carry audio information.

3.2.6 Audio Function Block: Independent part of an audio subunit that deals with audio-related functionality, and applies it to an audio channel cluster.

3.2.7 AV subunit: An instantiation of an entity that can be identified uniquely within an AV unit and offers a set of coherent functions.

3.2.8 AV unit: The physical instantiation of a consumer electronic device, e.g., a camcorder or a VCR, within a Serial Bus node. This document describes a command set that is part of the software unit architecture for AV units.

3.2.9 Byte: Eight bits of data.

3.2.10 GUID: A Globally Unique Identifier, similar to the EUI-64 (made globally unique by the context in which it is used), and used in this specification as a means to identify vendor or organizationally unique functionality

3.2.11 Fb-plug: A function block plug represents a connection point between function blocks.

3.2.12 Function block: An atomic piece of functionality represented by a number of data structures.

3.2.13 Isochronous: A term that indicates the essential characteristic of a time-scale or signal, such that the time intervals between consecutive instances either have the same duration or duration's that are integral multiples of the shortest duration. In the context of Serial Bus, "isochronous" is taken to mean a bounded worst-case latency for the transmission of data; physical and logical constraints that introduce jitter preclude the exact definition of "isochronous."

3.2.14 Plug: A physical or virtual end-point of connection implemented by an AV unit or subunit that may receive or transmit isochronous or other data. Plugs may be Serial Bus plugs, accessible through the PCR's; they may be external, physical plugs on the AV unit; or they may be internal virtual plugs implemented by the AV subunits.

3.2.15 RAC-ID: 24-bit company_ID obtained from the IEEE Registration Authority Committee (RAC)

3.2.16 Serial Bus: The physical interconnects and higher level protocols for the peer-to-peer transport of serial data, as defined by IEEE Std 1394–1995 and updates.

3.2.17 Stream: A time-ordered set of digital data originating from one source and terminating at zero or more sinks. A stream is characterized by bounded bandwidth requirements and by synchronization points, or time stamps, within the stream data.

3.2.18 Topology: In the context of this specification, a term used to denote an ordering relationship between function blocks. The ordering relationship is not necessarily a strict ordering relationship.

3.2.19 Unit architecture: The formal specification of the format and function of the software-visible resources and behaviors of a class of units. This document, in conjunction with the references above, defines a unit architecture for the class of AV devices.

3.3 Acronyms and Abbreviations

AV/C Audio Video Controller

IEEE: The Institute of Electrical and Electronics Engineers, Inc.

3.4 Unimplemented locations

3.4.1 Reserved locations

The reserved field shall be set as defined in section 3.1, "Rules for reserved fields", of "Enhancements to the AV/C General Specification 3.0, version 1.0.

4. Audio Function Topology and Control

4.1 Addressing in AV/C

To manipulate the physical properties of an audio AV function, its functionality must be divided into addressable entities. In the current AV/C Digital Interface Command Set the AV/C address is defined as (Subunit_type, Subunit_ID). The Subunit_type field is a 5 bit field providing for 32 types of subunits. This number can be increased by making use of a standard extension mechanism as defined in the general AV/C document. The Subunit_type for audio subunit is 01_{16}. The Subunit_ID is a 3 bit field. A mechanism is provided that enables the definition of a virtually unlimited number of subunit types and IDs, by using extension bytes. The patch model, used in connecting subunits, presents too complex a system to realize because it allows the possibility to connect any subunit to any other. In some cases subunits cannot physically be connected because either the path does not exist or the audio streams are incompatible.

Table 4.1 – Subunit_type for Audio Subunit

field	value
Subunit_type	01_{16}

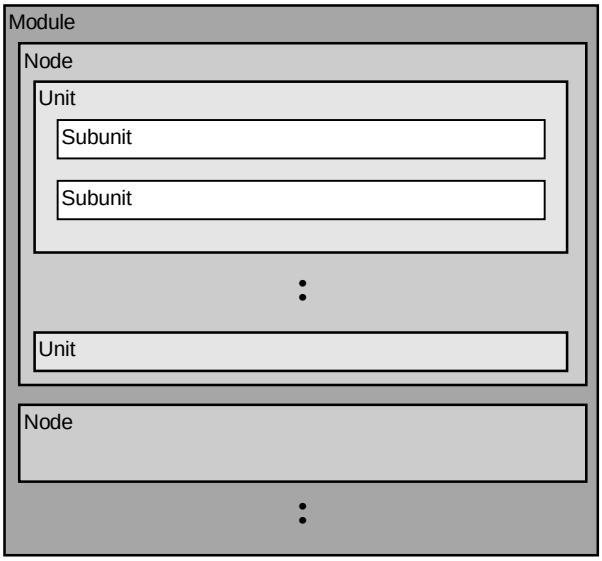

Figure 4.1 – Unit Architecture

Therefore, in this specification, the addressable entities describing the functionality of an audio subunit are called function blocks and they reside within an audio subunit. Connectivity within the audio subunit is handled through the selector and mixer model as opposed to the patch model which is used to connect subunits through isochronous plugs.

Also the selector model is more analogous to the implementation of audio equipment and function blocks can give detailed information inside a subunit if necessary.

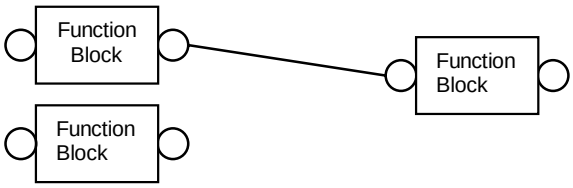

Figure 4.2 – The patch model (for comparison purposes only)

The selector model is chosen over the patch model to simplify the application design. It reduces the risk of making illegal connections between function blocks. Using the selector model in conjunction with configuration descriptors presents the application with all the legal possibilities supported by the device.

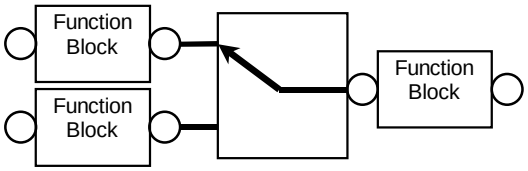

Figure 4.3 – The selector model

To enhance flexibility a device may support multiple predefined configurations that offer different connectivity between function blocks. In this way one could imagine a device containing hardware implementing a collection of function blocks that can be reconfigured using one of the predefined configurations, thus offering the user a truly wide choice of applicability.

4.2 The Audio Model

This specification uses a simple model to handle a wide range of possible audio processing equipment. This model is based on the following concepts:

1) This specification defines a new type of subunit, called the audio subunit. These subunits are composed of function blocks and are connected to the rest of the AV/C- and 1394-world through subunit plugs.

2) Audio data flowing into the subunit plugs are first pre-processed (unpacked, unframed, decoded) as necessary by the subunit plug and then distributed into the function block plugs. Similarly, signals flowing out of the subunit are collected onto a subunit plug in which all necessary formatting can take place. Audio format conversion may take place at a subunit plug or CODEC function block. Some subunits may not have CODEC function block although the subunit has CODEC functionality. In the case that CODEC function cannot be controlled, the CODEC may reside in a subunit plug.

3) Function blocks are connected through function block plugs (fb-plugs). These connections are not dynamic. Selector and mixer function blocks are used to establish paths through which audio data streams. Alternate connectivity can be achieved by using other predefined configurations.

4) Function blocks group a number of controls. Controls are to be thought of as representing physical properties of the function block that can be manipulated using the FUNCTION BLOCK command. To that end controls have a number of attributes associated with them. These attributes allow manipulating the control in absolute, relative or timed fashion.

5) A FUNCTION BLOCK command is defined to manipulate the attributes of these controls. As a general rule the ctype-field of the command is used to distinguish between the set and get form of the command.

6) In order to simplify the control of audio subunits across bus resets, all responses should be immediate. INTERIM responses should not be used.

4.3 Audio Subunit Plugs

The subunit plugs present on an audio subunit receive and transmit audio data. This audio data can have a variety of forms. They can be embedded in a DV audio-video stream, an MPEG program or transport stream or they can be in the form of an AM824 audio data stream.

Depending on the capabilities of the audio subunit, subunit plugs will be capable of processing these data streams. In this sense audio subunit plugs are very similar to CODEC function blocks. Inside the audio subunit it presents itself as an atomic entity having functionality. Outside the audio subunit it can be addressed and treated as defined in the general AV/C specification.

For example, a subunit plug capable of receiving an MPEG TS or PS will allow the selective extraction of one or more packetized elementary streams (PES) from the transport or program stream. The audio data present in the PES will be extracted and channeled to the proper fb-plugs.

Besides being embedded in a particular format of the data stream, the audio data can also be compressed (or encoded). The decoding of audio data is the responsibility of specialized function blocks called CODEC function blocks. CODEC functionality may exist in a subunit plug if it does not expose controls.

4.4 Audio Channel Clusters and Logical Channels

The fb-plugs carry audio data into the audio function blocks. The aggregate of audio data entering a fb-plug is called an audio channel cluster. An audio channel cluster is a group of logical audio channels that carries tightly related synchronous audio information. A stereo audio stream is a typical example of a two-channel audio channel cluster. Both the left and right signals of the stereo stream are logical audio channels. For example, a decoded MPEG-2 7+1 audio stream contains eight logical channels.

4.5 Function Blocks and Function Block Plugs

Function blocks provide the basic building blocks to describe most audio functions.

The model described in this specification serves to clarify the relationship between controls and to demonstrate the signal flow as it relates to this relationship. A designer does not necessarily have to reveal the inner workings of the audio subunit. In fact a lot of detail can remain hidden if desired. Only that part of the functionality needed for proper operation of the controls accessible by an outside agent needs to be revealed.

The level of detail of this description is at the discretion of the designer. As a general rule, the designer is expected to indicate the connectivity between function blocks. Only in cases where no ambiguity can exist as to the effect of a control inside a function block on the signals is he allowed to leave out the connection information. A typical example would be a simple audio subunit with a single volume control.

Audio functions are built by connecting together several of these function blocks through input and output function block plugs (fb-plugs). Audio function blocks have one or more input fb-plugs and one output fb-plug, where each fb-plug represents a cluster of logical audio channels inside the audio function block as

defined in the Consumer audio/video equipment – Digital interface – part 6: Audio and music data transmission protocol (see reference [R5]).

Function blocks are wired together by connecting their I/O fb-plugs according to the required topology by using selector or mixer subtype processing function blocks. This is different from subunits which are 'wired' together according to the patch model by using the AV/C CONNECT-command. The input fb-plugs of a function block are numbered starting from one up to the total number of input fb-plugs on the function block.

The information traveling between fb-plugs is not necessarily of a digital nature. It is possible to use the function block model to fully describe analog or even hybrid audio functions. The mere fact that I/O fb-plugs are connected together is a guarantee (by construction) that the protocol and format, used over these connections (analog or digital), is compatible on both ends.

Every function block in the audio function is fully described by its associated *function_block_dependent_information*. The *function_block_dependent_information* contains all necessary fields to identify the function block. It also provides the information allowing for the control of the function block.

4.6 Copy Protection

Because the audio functions are primarily dealing with digital audio streams, the issue of protecting these – often-copyrighted – streams can not be ignored. Therefore, this specification provides the means to preserve whatever copyright information is available. However, it is the responsibility of the controller to manage the flow of copy protection information throughout the audio subunit.

Copy protection issues come into play whenever digital audio streams enter or leave the audio subunit.

Therefore, the copy protection mechanism should preferably be implemented at the edge of the audio subunit.

Streams entering the audio subunit can be accompanied by specific information, indicating the copy protection level of that audio stream. Likewise, streams leaving the audio function should be accompanied by the appropriate copy protection information, if the hardware permits it. In these cases the audio subunit plugs should take care of copy protection issues.

In other cases copy control information may be embedded in the encoded audio data stream. In these cases it will be the responsibility of the relevant CODEC function block or a subunit plug that contains CODEC functionality to manage the copy control information properly

In the future, this specification may provide commands that can be used to manage the copy control information once a mechanism has been decided upon by the relevant working group(s).

4.7 Audio Function Blocks

This specification describes the following four different types of function blocks that are considered adequate to represent most audio functions available today and in the near future.

These are:

— Selector function block

— Feature function block

— Processing function block

— CODEC function block

The ensemble of function block dependent information provides a full description of the audio function. It should be possible to fully control the audio function with a controller. A control typically provides access to a specific physical audio property. Each control has a set of attributes that can be manipulated or that present additional information on the behavior of the control. Control attributes are manipulated using Function Block commands. Controls may be associated with the input and output audio channels they affect or may be master controls affecting some or all the audio channels in the function block depending on the implementation.

The master controls are cascaded **after** the individual channel controls. This setup is especially useful in multi-channel systems where the individual channel controls can be used for channel balancing and the master controls can be used for overall settings.

The logical channels in the cluster are numbered from one to the total number of channels in the cluster.

The 'master' channel has input and output channels numbered zero and is always present.

4.7.1 Selector Function Block

The selector function block (SFB) has n input fb-plugs and a single output fb-plug. It selects an input fb-plug from n input fb-plugs and routes an input audio channel cluster unaltered to the single output fb-plug.

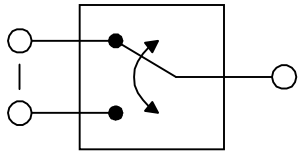

Figure 4.4 – Selector function block icon

4.7.2 Feature Function Block

The *feature function block* is essentially a multi-channel processing function block that provides basic manipulation of the incoming logical channels. For each logical channel, the *feature function block* optionally provides the following audio controls:

— Mute

— Volume

— Balance (Left/Right, Front/Rear)

— Tone Control (Bass, Mid, Treble)

— Graphic Equalizer

— Automatic Gain Control

— Delay

— Bass Boost

— Loudness

In addition, the *feature function block* can optionally implement controls that can influence all channels of the cluster at once for any of the controls listed above. In this way, 'master' controls can be implemented. The master controls are cascaded **after** the individual channel controls. This setup is especially useful in multi-channel systems where the individual channel controls can be used for channel balancing and the master controls can be used for overall settings.

The logical channels in the cluster are numbered from one to the total number of channels in the cluster.

The 'master' channel has channel number zero and is always present.

The *feature function_block_dependent_information* reports which controls are present for every channel in the *feature function block* and for the 'master' channel. All logical channels in a *feature function block* are fully independent. There shall be no cross couplings among channels within the *feature function block*. There will be as many logical output channels, as there are input channels. These are grouped into one audio channel cluster that enters the *feature function block* through a single input fb-plug and leaves the function block through a single output fb-plug.

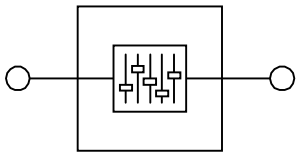

Figure 4.5 – Feature function block icon

4.7.3 Processing Function Block

The *processing function block* (PFB) represents a functional block inside the audio subunit that transforms a number of logical input channels, grouped into one or more audio channel clusters into a number of logical output channels, grouped into one audio channel cluster. Therefore, the *processing function block* can have multiple input fb-plugs and will have a single output fb-plug.

Processing function blocks are expected to support at least the *enable-processing control* if no other controls are present, allowing the software to bypass whatever functionality is incorporated in the *processing function block.*

This specification defines several standard transforms (algorithms) that are considered necessary to support additional audio functionality; these transforms are not covered by the other function block types but are commonplace enough to be included in this specification so that a generic controller can provide control for them.

FUNCTION_BLOCK command for the processing function block includes fields for *input audio channels* and *output audio channels*. For Audio Subunit Specification 1.0 only Mixer type Processing Function Blocks allow the *input audio channel number* to differ from the *output audio channel number*. Processing function blocks may also support "master" controls, which affect the master channels after the individual channel controls.

4.7.3.1 Mixer Processing Function Block

A subtype of the processing function block is the mixer. The *mixer processing function block* transforms a number of logical input channels into a number of logical output channels. The logical input channels are grouped into one or more audio channel clusters. Each cluster enters the *mixer processing function block* through an input fb-plug. The logical output channels are grouped into one audio channel cluster and leave the *mixer function block* through an output fb-plug.

Every input channel can be mixed into all of the output channels. If *n* is the total number of input channels and *m* is the number of output channels, then there are *n x m* mixing controls in the *mixer processing function block*.

Not all of these controls have to be physically implemented. Some controls can have a fixed setting and be non-programmable. The *function_block_type_dependent_information* in the *function_block_dependent_information* reports which of the controls are programmable in the *controls bitmap field*. Using this model, a permanent connection can be implemented by reporting the control as non-programmable and by returning a *control setting* of 0 dB when requested. Likewise, a missing connection can be implemented by reporting the Control as non-programmable and by returning a *control setting* of -∞dB.

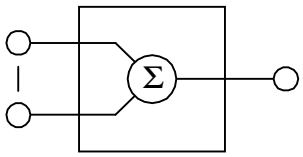

Figure 4.6 – Mixer Processing function block icon

Figure 4.7 gives a representation of two internal connections within a mixer. The horizontal line represents one input signal, while the vertical lines represent two outputs. The figure illustrates how an input signal can be distributed over several outputs. Similarly several inputs can be added with their respective weights into a single output.

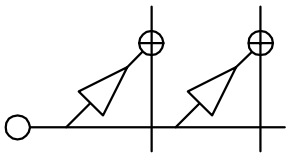

Figure 4.7 – Detail of a mixer connection

4.7.3.2 Generic Processing Function Block

The *generic processing function block* supports at least the enable-processing control. In addition it is identified by a GUID (globally unique identifier). This control can be used to extend the functionality presented in this specification. This functionality is identified by the GUID.

It allows for vendors and organizations to define specific function blocks to allow wider applicability of the concepts presented. This provides a clean upgrade path for this specification to keep up with technological developments in the field of audio control.

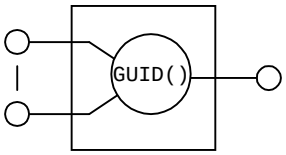

Figure 4.8 – Generic processing function block

4.7.3.3 Up/Down-mix Processing Function Block

The *up/down-mix processing function block* provides facilities to derive *m* output audio channels from *n* input audio channels. The algorithms and transforms applied to accomplish this are not defined by this specification and can be proprietary. The input channels are grouped into one input channel cluster that enters the processing function block over a single input fb-plug. Likewise, all output channels are grouped into one output channel cluster, leaving the processing function block over a single output fb-plug.

The *up/down-mix processing function block* can support multiple modes of operation (besides the bypass mode, controlled by the *enable-processing control*). The available input audio channels are dictated by the fb-plug to which the Up/Down-mix processing function block is connected. The *up/down-mix processing function block dependent information* reports which up/down-mixing modes the function block supports through its *Modes[] array*. Each element of the *Modes[] array* indicates which output channels in the output cluster are effectively used in a particular mode. The unused output channels in the output cluster must produce muted output. Mode selection is implemented using the FUNCTION BLOCK command.

Suppose the audio function's hardware is limited to reproducing only dual channel audio. Then the *up/down-mix processing function block* could use some (sophisticated) algorithms to down-mix the available spatial audio information into two ('enriched') channels so that the maximum spatial effects can be experienced, using only two channels (virtualization).

As a second example, suppose the hardware is capable of servicing eight discrete audio channels - for instance an MPEG-2 7.1 system. Now the *up/down-mix processing function block* could use certain techniques to derive meaningful content for the extra audio channels (Left of Center, Right of Center) that are present in the output cluster and are missing in the input channel cluster (such as the decoded PCM output of a 5.1 bitstream). This is a typical example of an up-mix situation.

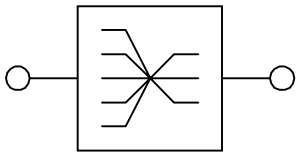

Figure 4.9 – Up/down-mix processing function block icon

4.7.3.4 Dolby Pro Logic Processing Function Block

The Dolby Pro Logic decoding process can be seen as an operator on the Left and Right logical channels of the input cluster of the function block. It is capable of extracting additional audio data (Center and/or Surround channels) from information that is transparently 'superimposed' on the Left and Right audio channels. It therefore differs from a true decoding process as defined for an Input Terminal. It can be applied on a logical audio stream anywhere in the audio subunit. The *Dolby Pro Logic processing function block* is a specialized derivative of the *Up/Down-mix processing function block*.

The *Dolby Pro Logic processing function block* can have the following multiple modes of operation (besides the bypass mode, controlled by the enable-processing control):

— Left, Right, Center channel decoding (Dolby 3 Stereo)

— Left, Right, Surround channel decoding

— Left, Right, Center, Surround decoding (Full Dolby Pro Logic)

Two additional variations are available that differ depending on whether a center speaker is available or not:

— No center speaker: phantom mode or no center

— Center speaker available: normal or wide mode.

The *Dolby Pro-Logic processing function block dependent information* reports which modes the function block supports. Mode selection is then implemented using the FUNCTION BLOCK command.

The Dolby Pro-Logic surround delay control is considered not to be part of the *Dolby Pro Logic Processing function block* and must be handled by a separate feature function block.

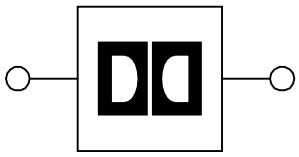

Figure 4.10 – Dolby Pro-Logic processing function block icon

4.7.3.5 3D-Stereo Extender Processing Function Block

The processing function block (3D-Stereo Extender) uses at least Left and Right logical input channels. It processes an existing stereo (two channels) soundtrack to add spaciousness and to make it appear to originate from outside the Left/Right speaker locations. Extended stereo effects can be achieved via various, straightforward methods. The algorithms and transforms applied to accomplish this are not defined by this specification and can be proprietary. The effects of the *3D-Stereo extender processing function block* can be bypassed at all times through manipulation of the enable-processing control. The size of the listening area (area in which the listener has to be placed with respect to speakers to hear the effect, also called sweet spot) can be controlled using the proper FUNCTION BLOCK command.

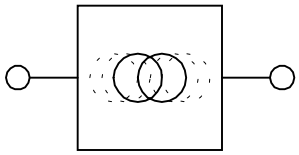

Figure 4.11 – 3D-stereo extender processing function block icon

4.7.3.6 Reverberation Processing Function Block

The *reverberation processing function block* is used to add reverberation effects to the original audio information.

These effects can range from small room reverberation effects to simulation of a large concert hall reverberation. A number of parameters can be manipulated to obtain the desired reverberation effects.

— Reverberation Type: Room1, Room2, Room3, Hall1, Hall2, Plate, Delay, and Panning Delay.

— Reverberation Level: sets the amount of reverberant sound.

— Reverberation Time: sets the time over which the reverberation will continue.

— Reverberation Early Reflection Time: Sets the time delay after which the first reflections arrive at the listener.

— Reverberation Delay Feedback: used with Reverberation Types Delay and Panning Delay. Sets the way in which delay repeats.

The effects of the *reverberation processing function block* can be bypassed at all times through manipulation of the enable-processing control.

In principal, the algorithm to produce the desired reverberation effect influences all channels as a whole.

It is entirely left to the designer how a certain reverberation effect is obtained. It is not the intention of this specification to precisely define all the parameters that influence the reverberation experience (for instance in a multi-channel system, it is possible to create very similar reverberation impressions, using different algorithms and parameter settings on all channels).

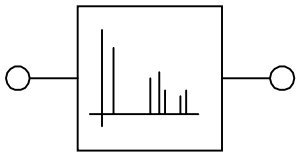

Figure 4.12 – Reverberation processing function block icon

4.7.3.7 Chorus Processing Function Block

The *chorus processing function block* is used to add chorus effects to the original audio information. A number of parameters can be manipulated in order to obtain the desired chorus effects.

— Chorus Level: controls the amount of the effect sound of chorus.

— Chorus Modulation Rate: sets the speed (frequency) of the modulator of the chorus.

— Chorus Modulation Depth: sets the depth at which the chorus sound is modulated.

The effects of the *chorus processing function block* can be bypassed at all times through manipulation of the enable-processing control.

In principle, the algorithm to produce the desired chorus effect influences all channels as a whole. It is entirely left to the designer how a certain chorus effect is obtained. It is not the intention of this specification to precisely define all the parameters that influence the chorus experience.

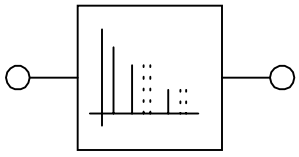

Figure 4.13 – Chorus processing function block icon

4.7.3.8 Dynamic Range Compressor Processing Function Block

The *dynamic range compressor processing function block* is used to intelligently limit the dynamic range of the original audio information. A number of parameters can be manipulated to influence the desired compression.

Compression ratio R: determines the slope of the static input-to-output transfer characteristic in the compressor's active input range. The compression is defined in terms of the compression ratio R, which is the inverse of the derivative of the output power PO as a function of the input power P_I when P_O and P_I are expressed in dB.

$$R^{-1} = \frac{\partial Log(P_O / P_R)}{\partial Log(P_I / P_R)}$$

P_R is the reference level and it is made equal to the so-called line level. All levels are expressed relative to the line level (0 dB), which is usually 15-20 dB below the maximum level. Compression is obtained when $R > 1$, $R = 1$ does not affect the signal and $R < 1$ gives rise to expansion.

— Maximum Amplitude: the upper boundary of the active input range, relative to the line level (0 dB). Expressed in dB.

— Threshold level: the lower boundary of the active input level, relative to the line level (0 dB).

— Attack Time: determines the response of the compressor as a function of time to a step in the input level. Expressed in ms.

— Release Time: relates to the recovery time of the gain of the compressor after a loud passage. Expressed in ms.

The effects of the *dynamic range compressor processing function block* can be bypassed at all times through manipulation of the enable-processing control.

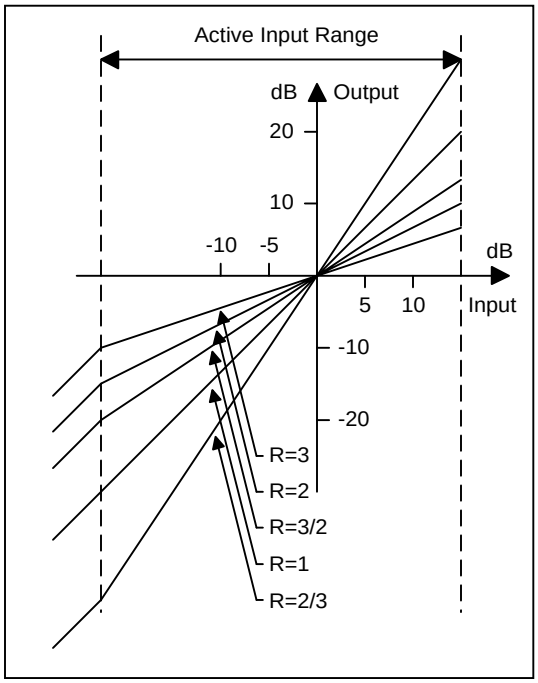

Figure 4.14 – Dynamic range compressor transfer characteristic

In principle, the algorithm to produce the desired dynamic range compression influences all channels as a whole. It is entirely left to the designer how a certain dynamic range compression is obtained.

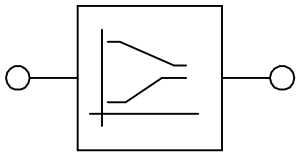

Figure 4.15 – Dynamic range compressor processing function block icon

4.7.4 The CODEC Function Block

The CODEC function block transforms non-linear encoded audio bit-streams into linear audio bit-streams. We assume that the subunit plug takes care of unpacking audio data from streams such as packetized elementary streams (PES) if the CODEC function block is connected to the subunit destination plug. Otherwise all the necessary processing is supposed to be done at the subunit plug.

A number of decoding function blocks are defined.

Decoders can range from very simple to very complicated. The simplest implementations may offer just baseline functionality, while more sophisticated versions may be able to detect the type of audio data that is streaming through them and switch decoding algorithms automatically.

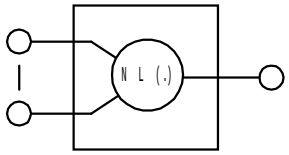

Figure 4.16 – CODEC function block

4.8 Control Attributes

A control can have the following attributes:

- CURRENT setting control attribute
- MINIMUM setting control attribute
- MAXIMUM setting control attribute
- RESOLUTION control attribute
- DURATION control attribute
- MOVE control attribute
- DELTA control attribute
- DEFAULT control attribute

As an example, consider a volume control inside a feature function block. By navigating through the descriptors and by issuing the appropriate FUNCTION BLOCK command, values for the volume control's attributes can be obtained and, for instance, used to correctly display the control's current state. Setting the volume control's current attribute allows one to change the setting of the volume control of the function block.

5. Audio Subunit Identifier Descriptor

The general subunit identifier structure is described in ref [R4]. The information and fields not detailed in that reference and of relevance to this specification are defined in sub-clauses 5.1 and following below. The information in the audio subunit identifier descriptor is static. The controller cannot modify the information by descriptor command such as WRITE DESCRIPTOR. An audio subunit that receives OPEN DESCRIPTOR command with subfunction WRITE OPEN addressed to audio subunit identifier descriptor shall return an AV/C response frame with response code "NOT IMPLEMENTED".

The audio subunit identifier descriptor contains the following information:

Table 5.1 – Audio subunit identifier descriptor

address offset	contents
00 00_{16}	descriptor_length
00 01_{16}	
00 02_{16}	generation_ID
00 03_{16}	size_of_list_ID
00 04_{16}	size_of_object_ID
00 05_{16}	size_of_object_position
00 06_{16}	number_of_root_object_lists (n)
00 07_{16}	
00 08_{16}	root_object_list_id_0
:	
:	:
:	root_object_list_id_n-1
:	
:	audio_subunit_dependent_length
:	
:	audio_subunit_dependent_information
:	
:	
:	manufacturer_dependent_length
:	
:	manufacturer_dependent_information
:	
:	

The *descriptor_length* field contains the number of bytes which follow in this descriptor structure. The value of this field does *not* include the length field itself.

The *generation_ID* field specifies which AV/C descriptor format is used by this subunit for all data structures it maintains, and the command sets which affect them. This field can have one of the following values:

Table 5.2 – Generation_ID values

Value	meaning
00_{16}	Data structure and command sets as specified in the AV/C General specification, version 3.0
01_{16}	Data structure and command sets as specified in the AV/C General specification, version 3.0 and the Enhancements to the AV/C General Specification 3.0, version 1.0
all others	Reserved for future specification

The *size_of_list_ID* field indicates the number of bytes used to indicate a list ID for this subunit. All lists maintained within the scope of this subunit shall use this number of bytes for their ID values.

The *size_of_object_ID* field specifies the number of bytes used for object_ID values managed by this subunit. All objects maintained within the scope of the subunit which have an ID should use this number of bytes for their ID. It is possible for some objects within the scope of a subunit to have ID values, and for some to not have ID values. If the subunit doesn't support object ID values for any of its objects, this field shall be set to 0.

The *size_of_object_position* field indicates the number of bytes used when referring to an object by its position in a list. All such references used within the subunit shall use this number of bytes for the position reference.

The size value of list ID, object ID, and object position shall be as follows.

Table 5.3 – The size value of list_ID, object_ID, and object_position

field name	value
size_of_list_ID	02_{16}
size_of_object_ID	00_{16}
size_of_object_position	02_{16}

The *root_object_list_id_x* fields are the ID values for each of the associated object lists.

The *number_of_root_object_lists* field indicates how many of these ID values are present.

The *audio_subunit_dependent_length* field specifies the number of bytes in the *audio_subunit_dependent_information* field.

The *audio_subunit_dependent_information* is defined in sub-clause 5.1.

The *manufacturer_dependent_length* and *manufacturer_dependent_information* fields are used for vendor-specific data. The format and contents are defined by the subunit manufacturer.

5.1 Audio Subunit Dependent Information

By audio subunit we mean any configuration of the audio function blocks whose associated data structures are defined later in this document.

The audio *subunit_dependent_information* contains the following information:

Table 5.4 – The audio subunit dependent information

address offset	Contents
00 00$_{16}$	audio_subunit_dependent_info_fields_length
00 01$_{16}$	
00 02$_{16}$	audio_subunit_version
00 03$_{16}$	number_of_configurations(p)
00 04$_{16}$	Configuration[1]_dependent_information
:	
:	
:	:
:	Configuration[p]_dependent_information
:	
:	

The *audio_subunit_dependent_info_fields_length* field specifies the number of bytes from the *audio_subunit_version* field through the last *configuration[]_dependent_information* field in the *audio_subunit_dependent_information* field. The value of this field does *not* include the length field itself.

The *audio_subunit_dependent_info_fields_length* is introduced so that extended fields can follow the existing fields, in case the audio subunit dependent information needs to be expanded in the future. Controllers can easily determine if any fields exist here by comparing the *audio_subunit_dependent_length* and *audio_subunit_dependent_info_fields_length* fields. In AV/C Audio Subunit Specification Version 1.0, the value of the *audio_subunit_dependent_info_fields_length* field is equal to the *audio_subunit_dependent_length* minus two. If the following formula is true:

audio_subunit_dependent_length > (*audio_subunit_dependent_info_fields_length* + 2)

then fields exist in this structure.

The *audio_subunit_version* field indicates the version number of audio subunit command specification that this audio subunit conforms to.

Table 5.5 – Audio subunit version

audio_subunit_version	Meaning
00_{16}	Version 1.0 of the audio subunit specification
all others	Reserved for future specification

The *number_of_configurations* and *configuration[]_dependent_information* fields are defined in sub-clause 5.1.1.

5.1.1 Configuration Dependent Information

The *number_of_configurations* field specifies how many configurations are supported in this audio subunit.

The *configuration[x]_dependent_information fields* each describes a configuration and topology of the function blocks in the audio subunit. They have the format as described in Table 5.6.

Table 5.6 – The configuration dependent information

address offset	contents
00 00_{16}	configuration[]_dependent_information_length
00 01_{16}	
00 02_{16}	configuration_ID
00 03_{16}	
00 04_{16}	master_cluster_information
:	
00 04_{16}+n-1	
00 04_{16}+n	number_of_subunit_source_plug_link_information (p)
00 05_{16}+n	subunit_source_plug[0]_link_information
00 06_{16}+n	
:	:
00 05_{16}+n+2*(p-1)	subunit_source_plug[p-1]_link_information
00 06_{16}+n+2*(p-1)	
00 07_{16}+n+2*(p-1)	number_of_function_block_dependent_information (m)
00 08_{16}+n+2*(p-1)	function_block_dependent_information[1]
:	
:	
:	:
:	function_block_dependent_information[m]
:	
:	

The *configuration[]_dependent_information_length* field specifies the number of bytes from the *configuration_ID* field through the last *function_block_dependent_information[]* field in the *configuration_dependent_information*. The value of this field does not include the length field itself.

The *configuration_ID* field specifies an ID of a configuration supported by an audio subunit.

The *master_cluster_information* is described in sub-clause 5.1.2.

The *number_of_subunit_source_plug_link_information* field specifies the number of the *subunit_source_plug[]_link_information* fields, which equals the number of audio subunit source plugs.

The *subunit_source_plug[]_link_information* is composed of *function_block_type* and *function_block_ID* fields.

Table 5.7 – subunit_source_plug_link_information

offset	contents
00 00$_{16}$	function_block_type
00 01$_{16}$	function_block_ID

The *function_block_type* field specifies the type of the function block or subunit destination plug to which the subunit source plug is connected. The value of F0$_{16}$ in this field indicates the subunit destination plug. The value of FE$_{16}$ in this field indicates that the subunit source plug is NOT connected.

The *function_block_ID* field specifies the ID of the function block or subunit destination plug number to which the subunit source plug is connected.

The *number_of_function_block_dependent_information* field specifies the number of the *function_block_dependent_information* fields, which is equal to the number of function blocks in the audio subunit.

The *function_block_dependent_information* data structures are described in clause 9.

5.1.2 Cluster Information

A Cluster information describes the characteristics of an audio channel cluster, which is a group of logical audio channels.

All logical audio channels which are present in a configuration are specified in each *configuration[]_dependent_information* fields of a subunit Identifier descriptor. This is specified in the *master_cluster_information* field. Each *configuration[]_dependent_information* field contains the *master_cluster_information* field.

Also, according to the information of the *master_cluster_information* field, the logical audio channels in each function block are specified in each *function_block_dependent_information*. This is specified in *cluster_information* field.

The *master_cluster_information* and *cluster_information* fields include the *ChConfigType* field. The *ChConfigType* field indicates the structure on which channel configuration field is based. The following table shows the currently defined structures.

Currently, only one structure type is defined: GENERIC_MULTI-SPEAKER

There are two special cases. One is that all logical audio channels in a function block are the same as that in *master_cluster_information* field. Then, the value of *ChConfigType* field is 00_{16}. Another is that a controller must trace the connection ‘upstream’. Then, the value of *ChConfigType* field is 01_{16}. The following summarizes the value of *ChConfigType* field.

Table 5.8 – The value of ChConfigType

Value	Meaning
00_{16}	Same as master_cluster_information
01_{16}	Same as upstream
02_{16}	GENERIC_MULTI-SPEAKER
03_{16} - FF_{16}	Reserved for future expansion

The value of *ChConfigType* field in *master_cluster_information* field shall not be set to 00_{16} or 01_{16}.

Logical audio channel numbering in the cluster starts with channel one up to the number of channels in the cluster.

The virtual channel zero is used to address a Master Control in a function block, effectively influencing the master channel controls. The master channel controls are defined depending upon the type (and sub-type) of the function block and the *control_selector* of FUNCTION BLOCK command, and are indicated the capabilities in the *function_block_dependent_information* field of the *audio_subunit_dependent_information*. The maximum number of independent channels in an audio cluster is limited to 253. Indeed, Channel zero is used to reference the master channel and code FF_{16} (255) is used in function block commands to indicate that the parameters hold values for all available addressed controls. This way all channels can be operated on simultaneously, with individual values for each individual channel. Code FE_{16} (254) is used to in FUNCTION_BLOCK commands to indicate “VOID” channel. For example, it is used in FUNCTION_BLOCK commands to the processing function block when this function block does not support an input channel number.

5.1.2.1 GENERIC_MULTI-SPEAKER

When the value of *ChConfigType* is GENERIC_MULTI-SPEAKER, each channel in the audio cluster may be tied to a certain speaker location in the listening space. A trivial example of this is a cluster that contains Left and Right logical audio channels.

To be able to describe more complex cases in a manageable fashion, this specification proposes a conventional ordering of logical channels in an audio channel cluster.

The cluster information allows descriptions of channel orderings. In case the conventional ordering of channels is followed, a shorter cluster information is possible.

This method is offered to allow the handling of special channel ordering in the future.

There are fifteen predefined spatial locations:

— Left Front (L)

— Right Front (R)

— Center Front (C)

— Low Frequency Enhancement (LFE) [Super woofer]

— Left Surround (L_S)

— Right Surround (R_S)

- Left of Center (L_C) [in front]
- Right of Center (R_C) [in front]
- Surround (S) [rear]
- Side Left (S_L) [left wall]
- Side Right (S_R) [right wall]
- Top (T) [overhead]
- Bottom (B) [bottom]
- Left Front Effect [FEL]
- Right Front Effect [FER]

If there are logical channels present in the audio cluster that correspond to some of the previously defined speaker positions, then they should preferably appear in the order specified in the above list.

To characterize an audio cluster, a cluster information is introduced. This information is embedded within the *master_cluster_information* and *cluster_informatio*n. The *master_cluster_information* is contained in each *configuration[]_dependent_information* of each configuration. The *cluster_information* is contained in the *function_block_dependent_information* of some function block.

These are the types of function blocks that potentially affect the number and spatial location of the logical audio channels in one of the input function block plugs and therefore a cluster information is associated with the output function block plug of these function blocks.

The cluster information contains the following fields:

- *number_of_channels*: a number that specifies how many logical audio channels are present in the cluster.
- *predefined_ChannelConfig*: an array of bit fields that indicates which spatial locations are present in the cluster.

The bit allocations of the *predefined_ChannelConfig* field are as follows:

- Bit 0: Left Front (L)
- Bit 1: Right Front (R)
- Bit 2: Center Front (C)
- Bit 3: Low Frequency Enhancement (LFE)
- Bit 4: Left Surround (L_S)
- Bit 5: Right Surround (R_S)
- Bit 6: Left of Center (L_C)
- Bit 7: Right of Center (R_C)
- Bit 8: Surround (S)
- Bit 9: Side Left (S_L)
- Bit 10: Side Right (S_R)
- Bit 11: Top (T)
- Bit 12: Bottom (B)
- Bit 13: Left Front Effect (FEL)

— Bit 14: Right Front Effect (FER)

— Bit 15: Non-conventional ordering

Different cases must be considered depending on the value of bit 15 of the *predefined_ChannelConfig* field.

Table 5.9 – master_cluster_information (GENERIC_MULTI-SPEAKER) shows the format of *master_cluster_information* when the value of *ChConfigType* is GENERIC_MULTI-SPEAKER.

Table 5.9 – master_cluster_information (GENERIC_MULTI-SPEAKER)

address offset	Contents
00 00$_{16}$	master_cluster_information_length
00 02$_{16}$	number_of_channels (n)
00 03$_{16}$	ChConfigType
00 04$_{16}$	Predefined_ChannelConfig
00 05$_{16}$	
00 06$_{16}$	ChannelNames[1]
00 06$_{16}$ + 1	
:	:
00 06$_{16}$+2*(n-1)	ChannelNames[n]
00 06$_{16}$+2*(n-1)+1	

Table 5.10 shows the format of *cluster_information* for each *function_block_dependent_information* when the value of *ChConfigType* in *master_cluster_information* field is GENERIC_MULTI-SPEAKER.

Table 5.10 – cluster_information in function_block_dependent_information

address offset	Contents
00 00$_{16}$	number_of_channels (n)
00 01$_{16}$	ChConfigType
00 02$_{16}$	predefined_ChannelConfig
00 03$_{16}$	
00 04$_{16}$	non-predefined_ChannelConfig
:	

When the value of the ChConfigType is 00_{16} or 01_{16}, the predefined_ChannelConfig field and non-predefined_ChannelConfig fields do not exist.

The *non-predefined_ChannelConfig* field is a bitmap field indicating non-predefined channels. Each bit in the *non-predefined_ChannelConfig* field shows each non-predefined channel from msb in the order specified in the *ChannelNames[]* fields of the *master_cluster_information* field.

5.1.2.2 Conventional Ordering

Each bit set in the *predefined_ChannelConfig* bit map indicates there is a logical channel in the cluster that carries audio information, destined for the indicated spatial location. The channel ordering of the logical audio channels in the cluster must correspond to the ordering, imposed by the above list of predefined spatial locations. When conventional ordering is used, the last bit of the ChannelConfig field is set to 0, and the channel cluster information has the structure shown in Table 5.11.

If there are more channels in the cluster than there are bits set in the *predefined_ChannelConfig* field, (i.e. *number_of_channels* > [Number_Of_Bits_Set]), then the channels indicated by bits set in the *predefined_ChannelConfig* field (i.e. first [Number_Of_Bits_Set] channels) take the spatial positions**.** The remaining channels in the cluster have 'non-predefined' spatial positions (positions that do not appear in the *predefined_ChannelConfig* field).

If none of the bits in *predefined_ChannelConfig* are set, then all channels have non-predefined spatial positions.

If one or more channels have non-predefined spatial positions, their spatial location description can optionally be derived from the *ChannelNames* field.

ChannelNames: index of a text database object in a text database list that describes the name of each logical channel in the cluster.

address offset	msb							lsb
00 01_{16}	x	X	X	x	x	x	X	x
00 02_{16}	x	X	X	x	x	x	X	0

Figure 5.1 – The predefined_ChannelConfig field

Table 5.11 – The Channel Cluster information for conventional ordering

address offset	contents	Description
00 00_{16}	number_of_channels (n)	Number of logical output channels in the cluster.
00 01_{16}	ChConfigType	Specifies channels configuration type
00 02_{16}	predefined_ChannelConfig	Describes the spatial location of the logical channels in the cluster.
00 04_{16}	ChannelNames[1]	Entry Number of a Text Object in a Text Database, describing the name of the first logical channel in the cluster.
…	…	…
00 04_{16}+2*(n-1)	ChannelNames[n]	Entry Number of a Text Object in a Text Database, describing the name of the last logical channel in the cluster.

5.1.2.3 Non-conventional ordering

Audio channel clusters that do not follow the conventional ordering, either because a different ordering scheme is used, or some of the channels are missing, bit 15 of the *ChannelConfig* field will be set to indicate an extension to the field. The fields following and including the *ChannelConfig* field must, in this case be interpreted as indicating which conventional location corresponds to the channel number (numbering always starts from 1). When non-conventional ordering is used, the last bit of every ChannelConfig field is set to 1, and the channel cluster information has the structure shown in Table 5.12.

address Offset	msb							lsb
00 01_{16}	x	X	X	x	x	x	X	x
00 02_{16}	x	X	X	x	x	x	X	1

Figure 5.2 – The ChannelConfig field

Table 5.12 – The Channel Cluster information for non-conventional ordering

address offset	contents	Description
00 00_{16}	number_of_channels (n)	Number of logical output channels in the Processing Function block's output channel cluster.
00 01_{16}	ChConfigType	Specifies channels configuration type
00 02_{16}	ChannelConfig[1]	Describes the spatial location of the logical channels in the Function block's audio channel cluster.
00 $02+2_{16}$	ChannelConfig[2]	
…	…	
00 02_{16}+2*(n-1)	ChannelConfig[n]	
00 02_{16}+2*n	ChannelNames[1]	Entry Number of a Text Object in a Text Database, describing the name of the first logical channel in the Processing Function block's audio channel clusters.
…	…	…
00 02_{16}+2*n+2*(n-1)	ChannelNames[n]	Entry Number of a Text Object in a Text Database, describing the name of the last logical channel in the Processing Function block's audio channel cluster.

6. Audio Subunit Objects

All AV/C audio subunit objects use the general AV/C object format as described in the references 'AV/C Digital Interface Command Set General Specification Ver3.0'. The important information for this specification is the *entry_specific_information*, which is described for each audio subunit object. The reader is encouraged to review the other reference material for an overall understanding of how these data structures fit into the AV/C object and object list model.

The general AV/C object descriptor uses one byte for the *entry_type* field, which describes the object. In order to provide additional information which describes the exact nature of the object, each audio subunit object contains additional information.

6.1 Audio Subunit Object Types

The following basic *entry_type* values are defined for the audio subunit model:

Table 6.1 – Audio subunit object entry_type definitions

Entry Type	Value	Meaning
Child Directory Object	90_{16}	This object holds the child list ID of a Child Contents List, used to construct the hierarchical file system for subunits which support it.
Text Database Object	93_{16}	This object represents a small piece of text, such as the information of a function block.
------	all other values in the subunit-specific range	

These two entry types are the same as the entry types defined for the disc subunit model. For more details, please refer to the document titled 'AV/C Disc Subunit General Specification ver 1.0'.

7. Audio Subunit Object Lists

REMINDER: In order to fully understand the information presented in this sub-clause, it is necessary to understand the general AV/C object and object list concepts which are described in reference 'AV/C Digital Interface Command Set General Specification Ver3.0'.

7.1 Audio Subunit List Types

The following *list_type* value is defined for the AV/C audio subunit. This *list_type* is the same as the *list_type* defined for the disc subunit model. The table provides a brief description of the lists, but further details are provided in 'Clause 9 of AV/C Disc Subunit General Specification version 1.0'

Table 7.1 – audio subunit list type

list name	list_type	comments
Text Database List	86_{16}	The list contains text database objects
-----	all others in the subunit specific range	Reserved for future specification

7.2 Text Database List Hierarchy Structure

Supporting text database lists is optional. If supported, there shall be one text database list as a root list and some child text database lists. The ID value of the root text database list is found in the audio subunit identifier descriptor. In the audio subunit, ID value of the root text database list should be 1800_{16}. The root text database list shall contain one or more child directory objects, each of which points to a child text database list. The root list shall not contain text database objects. Each child text database list holds text information regarding to audio subunit identifier descriptor.

Each text database objects in one text database list should have same character code and language code. The whole structure of text database list is shown below.

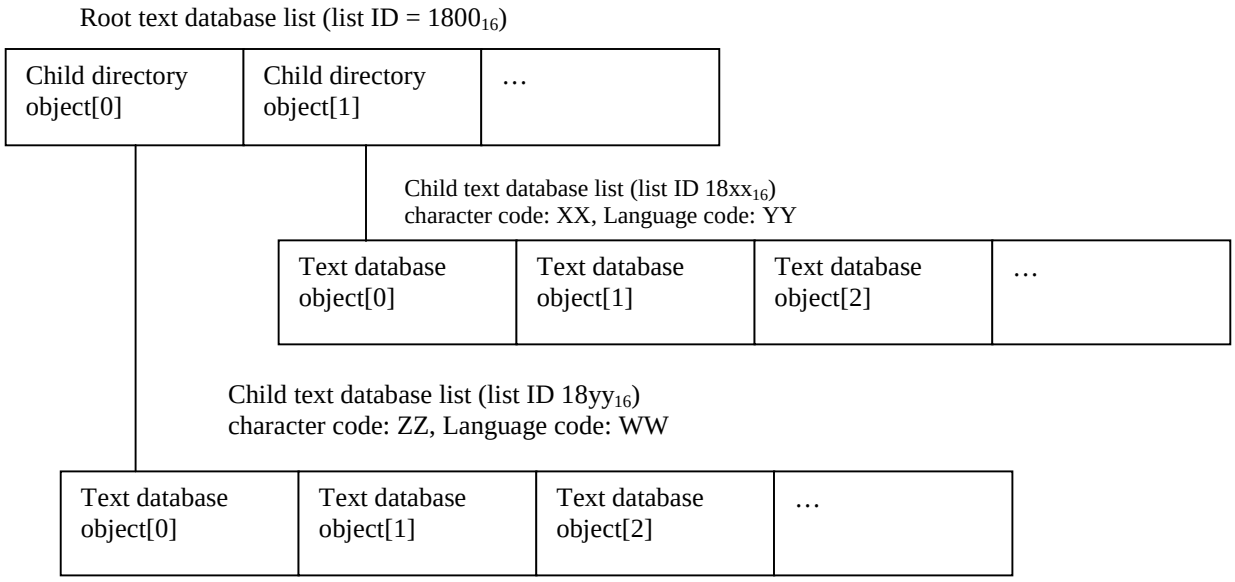

Figure 7.1 – Example of the whole structure of text database lists

Each text database object which has same object position but in the different list holds same contents describing in the different character code and/or language code. (For example, object_ entry[0]'s of all child text database lists contain text information about the Selector Function Block[0]). When an object entry in particular list has no text data, it should be an empty object entry.

The *character_ code_ info_ block, language_ code_ info_ block* which describe the character code, language code for the text database object may be included in the text database object entry_ specific_ information. There shall be exactly one of each of these info blocks, OR there may not be; in the latter case, the default text format is minimal English ASCII as noted in the *raw_ text_ info_ block* description.(Please see 'Enhancement to the AV/C General Specification 3.0' Sub-clause 6.8 'Raw Text Info Block').

One text database object may be pointed by some indexes in the audio subunit identifier descriptor.

The list ID allocation for each text database list is shown.

Table 7.2 – Text Database List ID allocation

list_type	content		list_ID
Text Database List	Root		1800_{16}
	Child	minimal English ASCII	1801_{16}
		other text database list	1802_{16}-$18FF_{16}$

Supporting child text database list with list_ID = 1801_{16} (minimal English ASCII) is optional.

For this version of AV/C Audio Subunit, a controller can not modify the information in the text database lists. An audio subunit that receives OPEN DESCRIPTOR command with subfunction WRITE OPEN addressed to text database list or descriptor in the text database list shall return an AV/C response frame with response code "NOT IMPLEMENTED".

7.3 Text Database List Reference from Audio subunit identifier descriptor

The following diagram illustrates the field referencing text database list from the audio subunit identifier descriptor:

Table 7.3 – The fields referencing text database list

Address offset	contents
00 00_{16}	object_position
00 01_{16}	

The *object_position* field indicates the position of the target text database object, within the text database list. The number of bytes in this field is determined by the *size_of_object_position* field of the subunit identifier descriptor. If all bits of this field are "1", then the field does not refer to a text database list. This means no text information is available for this field. If text database lists is NOT supported by an audio subunit, all bits of the fields which reference text database list should be "1".

8. Function Block Dependent Information

8.1 Common Function Block Dependent Information

A function block is uniquely identified by the values in the *function_block_type* and the *function_block_ID* fields of the *function_block_dependent_information*. These values must be passed in the corresponding fields of each command that is directed to the function block.

The general *function_block_dependent_information* contains the following information:

Table 8.1 – General function block dependent information

offset	contents	description
00 00_{16}	function_block_dependent_information_length	Size of this function_block_dependent_information in bytes. The value of this field does not include the length field itself
00 01_{16}		
00 02_{16}	function_block_type	Constant identifying the type of function block within the audio subunit.
00 03_{16}	function_block_ID	Constant together with the previous field uniquely identifying the function block within the audio subunit. Both constants are used in all commands to address this a function block.
00 04_{16}	function_block_name	Index of a text database object in a text database list describing this function block
00 05_{16}		
00 06_{16}	number_of_input_fb_plugs (p)	Number of input fb-plugs of this function block: p
00 07_{16}	source_ID[0]	ID and type of the function block or plug to which the first input plug[0] of this function block is connected.
00 08_{16}		
:	:	:
00 07_{16}+2*(p-1)	source_ID[p-1]	ID and type of the function block or plug to which the last input plug[p-1] of this function block is connected.
00 08_{16}+2*(p-1)		
00 09_{16}+2*(p-1)	cluster_information_length	Length in bytes of the *cluster_information* field. The value of this field does not include the length field itself. This field is 2 bytes in size.
:		
:	cluster_information	Details of cluster in this function block
:		
:		
:	function_block_type_dependent_information_length	Length in bytes of the *function_block_type_dependent_information* field. The value of this field does not include the length field itself. This field is 2 bytes in size.
:		
:	function_block_type_dependent_information	Defined by each *function_block_type.*
:		
:		

The *function_block_name* field specifies an index of a text data object in a text database list that further describes the function block.

The *number_of_input_fb_plugs* field contains the number of input fb-plugs of the function block.

The connectivity of the input fb-plugs is described via the *source_ID[]* fields array, containing p elements. The index i into the array is zero-based and directly related to the input fb-plug numbers. The *source_ID[i]* fields contains the ID and type of the function block or subunit destination plug to which input fb-plug i is connected.

The *source_ID* is composed of *function_block_type* (*subunit_destination_plug*) and *function_block_ID* (*subunit_destination_plug_number*).

Table 8.2 – Source_ID field

offset	contents
$00\ 00_{16}$	function_block_type
$00\ 01_{16}$	function_block_ID

The *function_block_type* field specifies the type of the function block or subunit destination plug to which the function block is connected via its input fb-plug. The value of $F0_{16}$ in this field indicates the subunit destination plug. The value of FE_{16} in this field indicates that the input fb-plug is NOT connected.

8.2 Selector Function Block Dependent Information

The selector function block is uniquely identified by the values in the *function_block_type* and *function_block_ID* fields of the selector *function_block_dependent_information*. These values shall be passed in the corresponding fields of each command that is directed to the selector function block. The selector function block does not have any *cluster_information* and *function_block_type_dependent_information*. The *cluster_information_length* and *function_block_type_dependent_information_length* are zero.

8.3 Feature Function Block Dependent Information

The *feature function block* is uniquely identified by the value in the *function_block_type* and *function_block_ID* fields of the *feature_function_block_dependent_information*.

The *source_ID* field is used to describe the connectivity for this *feature function block*. It contains the ID and type of the function block or plug to which this feature function block is connected via its input fb-plug. The cluster information, describing the logical channels entering the feature function block is not repeated here. It is up to the controller software to trace the connection ‘upstream’ in order to locate the cluster information pertaining to this audio channel cluster.

The *Controls[]* array is an array of bit maps, each indicating the availability of certain audio controls for a specific logical channel or for the master channel 0. For future expandability, the number of bytes occupied by each element (n) of the *Controls[]* array is indicated in the *size_of_controls* field. The number of logical channels in the cluster is denoted by ch. This number can be derived from the *cluster_information* field of Table 8.1.

The feature *function_block_type_dependent_information* field contains the following information.

Table 8.3 – Feature function_block_type_dependent information

offset	contents	description
00 00_{16}	Controls_specific_information_length	Total length in bytes of the size_of_controls field, and the general_tag field and Controls[] fields.
00 02_{16}	size_of_controls	Size in bytes of an element of the Controls[] array: n
00 04_{16}	general_tag	Bit 0…1: Volume tag bit_field indicating the purpose of the volume 0: general volume 1: master volume 2: input trim 3: output trim Bit 2…7: reserved as bit field for future use.
00 05_{16} : : : : :	Controls[0]	A bit set to 1 indicates that the mentioned control is supported for master channel 0: Bit 0: Mute Control Bit 1: Volume Control Bit 2: LR Balance Control Bit 3: FR Balance Control Bit 4: Bass Control Bit 5: Mid Control Bit 6: Treble Control Bit 7: Graphic Equalizer Control Bit 8: Automatic Gain Control Bit 9: Delay Control Bit 10: Bass Boost Control Bit 11 Loudness Control Bit 12...(n*8-1): Reserved
00 05_{16}+n	Controls[1]	A bit set to 1 indicates that the mentioned control is supported for logical channel 1.
…	…	…
00 05_{16}+(ch*n)	Controls[ch]	A bit set to 1 indicates that the mentioned control is supported for logical channel ch.

8.4 Processing Function block Dependent Information

8.4.1 Mixer Dependent Information

The mixer processing function block is uniquely identified by the values in the function_block_type and the function_block_ID and process_Type fields of the processing function block dependent information.

The *number_of_input_fb_plugs* field contains the number of input fb-plugs (p) of the processing function block. This equals the number of audio channel clusters that enter the processing function block.

As mentioned before, every input audio channel can virtually be mixed into all of the output audio channels. If n is the total number of logical input channels, contained in all the audio channel clusters, entering the mixer function block, and m is the number of logical output channels, then there are n x m mixing controls in the mixer function block, some of which may not be programmable.

The processing function block dependent information reports which controls are programmable in the controls bitmap field. This bitmap must be interpreted as a two-dimensional bit array that has a row for each logical input channel and a column for each logical output channel. If a bit at position [u, v] is set, this means that the mixer processing function block contains a programmable mixing control that connects input channel u to output channel v. If bit [u, v] is clear, this indicates that the connection between input channel u and output channel v is non-programmable. Its fixed value can be retrieved through the appropriate command. The valid range for u is from one to n. The valid range for v is from one to m.

As an example, we illustrate a mixer processing function block that has 3 input function-block plugs and 8 logical input channels and 4 output channels. This could be illustrated as follows:

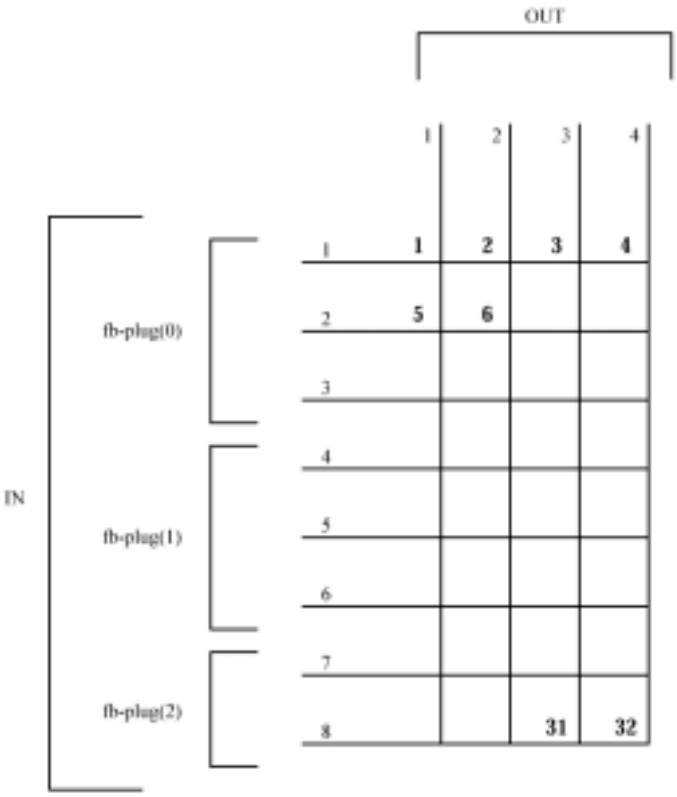

Figure 8.1 – Example of 8-in, 4-out mixer

The *controls bitmap* field would appear as below:

msb							lsb
1	2	3	+	+			
				+	+	31	32

Figure 8.2 – Example *controls bitmap field*

The *controls field* stores the bit array row after row where the msb of the first byte corresponds to the connection between input channel 1 and output channel 1. If (n x m) is not an integer multiple of 8, the bit array is padded with zeros until an integer number of bytes is occupied. The number of bytes used to store the bit array, N, can be calculated as follows:

```
IF ((n x m) MOD 8) <> 0 THEN
{ N = ((n x m) DIV 8) + 1}
ELSE
{N = ((n x m) DIV 8)}
```

The next table details the structure of the processing *function_block_type_dependent_information (mixer)* in the processing *function_block_dependent_information* (mixer):

Table 8.4 – Processing function_block_type_dependent_information (mixer)

address offset	contents	description
00 00_{16}	process_type (mixer)	identifier of sub-type of processing
00 01_{16}	size_of_controls (N)	Size in bytes of the Controls-field
00 02_{16}		
00 03_{16}	Controls	The bitmap indicating which mixing controls are programmable.
:		

Since a *processing function block* can redefine the spatial locations of the logical output channels, contained in its output cluster (its output fb-plug), there is a need for a *cluster information*. This descriptor will characterize the cluster that leaves the *processing function block* over the single output fb-plug ('downstream' connection).

8.4.2 Generic Processing Dependent Information

The generic processing function block provides for extensibility. By using the mechanism of a GUID, it becomes possible to define a new type of functionality. The GUID is based on the RAC_ID as defined in reference [R1].

The 24-bit RAC_ID field uniquely identifies the organization or vendor responsible for the definition of the processing function block; its value shall be assigned by the IEEE/RAC. Unique identifiers may be obtained from[1]:

Institute of Electrical and Electronic Engineers, Inc.
Registration Authority Committee
445 Hoes Lane
Piscataway, NJ 08855-1331

The RAC_ID is extended by a 40-bit number assigned by the owner of the RAC_ID to uniquely identify the functionality of a particular generic function block.

[1] The RAC_ID field is variously referred to in ISO/IEC 13213:1994 as an organizationally unique ID (OUI), company ID or vendor ID. Because of confusion about byte ordering within this 24-bit number, this standard renames the field to avoid any reference to earlier, conflicting definitions.

Table 8.5 – Generic processing block dependent information

address offset	contents	description
00 00_{16}	process_type	identifier of sub-type of processing
00 01_{16}	size_of_controls	Size in bytes of the Controls field: n
00 03_{16} … 00 03_{16}+n-1	Controls	A bit set to 1 indicates that the mentioned control is supported: Bit 0: Enable Processing Bit 1: Mode Select Bit 2..(n*8-1): GUID specific
00 03_{16}+n … 00 03_{16}+n+7	GUID	The GUID used to identify the object that interacts with this processing function block
00 03_{16}+n+8	GUID_dependent_information_length	total length of the GUID dependent datastructure
00 03_{16}+n+10	GUID dependent information	This data structure is determined by the owner of the GUID

8.4.3 Up/Down-mix Processing Dependent Information

The *process_type* field of the common *processing function block* dependent information contains the value UP/DOWNMIX_PROCESS. (The *Up/Down-mix processing function block* has a single input fb-plug. Therefore, the *number_of_inputs* field shall contain the value 1).

The *Controls* field is a bitmap, indicating the availability of certain audio Controls in the *Up/Down-mix processing function block*. For future expandability, the number of bytes occupied by the *Controls* field is indicated in the *size_of_controls* field.

Bit 0 of the *Controls* field represents the Enable Processing Control. The Mode Select Control (Bit 1) is used to change the behavior of the *processing function block* by selecting different modes of operation.

The *Up/Down-mix processing function dependent information* of the *Up/Down-mix processing function block* describes the supported modes of operation of the *processing function block*. Selecting a mode of operation is done by manipulating the *Mode* control. The number of supported modes (*m*) is contained in the *number_of_modes* field. This field is followed by an array of mode fields, *Modes[]*. The index *I* into this array is zero-based and directly related to the number of the mode described by entry *Modes[I]*. It is the value *I* that must be used as a parameter for the audio command to select or retrieve the mode *i*.

The bit allocations in the *Modes[]* fields are very similar to those of the *predefined_ChannelConfig* field in a *cluster information* i.e. a bit set in the *Modes[I]* field indicates that for mode *I*, the *Up/Down-mix processing function block* produces meaningful audio data for the logical channel that is associated with the position of the set bit. Logical channels that are present in the output cluster but are not used in a certain mode are considered to be inactive and at most produce silence in that mode.

Each *Modes[I]* field can only contain set bits for those logical channels that are present in the output channel cluster. In other words, all *Modes[]* fields can only contain a subset of the *predefined_ChannelConfig* field of the *cluster information* of the function block. Furthermore, logical channels that have a non-predefined spatial position can not be marked as active in the *Modes[]* fields. It is therefore assumed by default that they are active.

The following table describes th *Up/Down-mix processing function block dependent information*.

Table 8.6 – Up/Down-mix processing function dependent information

address offset	contents	description
00 00_{16}	process_type (up/down mix)	identifier of sub-type processing.
00 01_{16}	size_of_controls	Size in bytes of the Controls field: n
00 02_{16}		
00 03_{16} … 00 03_{16}+n-1	Controls	A bit set to 1 indicates that the mentioned control is supported: Bit 0: Enable Processing Bit 1: Mode Select Bit 2..(n*8-1): Reserved for Processing specific
00 03_{16}+n	number_of_modes	Number of modes, supported by this Processing function block: m
00 04_{16}+n	size_of_modes (p)	Defines the size of the Modes Field in bytes.
00 05_{16}+n	Modes[0]	Describes the active logical channels in mode 0.
:		
…	…	…
00 05_{16}+n+ ((m-1) * p)	Modes[m-1]	Describes the active logical channels in mode m-1.
:		

The *size_of_controls* field equals 1.

8.4.4 Dolby Pro Logic Processing Dependent Information

The *process_type* field of the common Processing function block descriptor contains the value DOLBY_PRO_LOGIC. This value can be found in Table A.2 – Processing type encoding.

The Dolby Pro Logic Processing function block has a single input fb-plug. Therefore, the *number_of_inputs_plugs* field must contain the value 1.

Bit 0 of the *Controls* field represents the Enable Processing Control. The Mode Select Control (Bit 1) is used to change the behavior of the *processing function block* by selecting different modes of operation.

Although the input cluster may contain a number of logical channels, the Dolby Pro Logic Processing Function Block only uses Left and Right logical input channels as input for the decoding process. Obviously, these two logical channels must be present in the input cluster for the function block to operate properly. All other logical channels are discarded.

The output cluster may contain logical channels other than Left, Right, Center, and/or Surround (these must be present) to facilitate connectivity within the audio function. Channels that are present in the output cluster but do not participate in the chosen mode of operation must be muted.

The process-specific descriptor of the *processing function block (Dolby Pro Logic)* describes the supported modes of operation of the Processing Function Block. The number of supported modes (*m*) is contained in the *number_of_modes* field. This field is followed by an array of mode fields, *Modes[]*. The bit allocations in the *Modes[]* fields are very similar to those of the *predefined_ChannelConfig* field in a cluster information i.e., a bit set in the *Modes[j]* field indicates that for mode *j*, the *processing functon block (Dolby Pro Logic)* produces meaningful audio data for the logical channel that is associated with the position of the set bit.

The *processing function block (Dolby Pro Logic)* supports the following different modes:

Table 8.7 – Dolby Modes

Contents	description
Left, Right, Normal Center channel decoding **Modes[]** = 0007_{16}	(Dolby 3 stereo Normal)
Left, Right, Wide Center channel decoding **Modes[]** = 0807_{16}	(Dolby 3 stereo Wide)
Left, Right, Surround channel decoding **Modes[]** = 0103_{16}	(Dolby Pro Logic Phantom)
Left, Right, Normal Center, Surround decoding **Modes[]** = 0107_{16}	(Dolby Pro Logic Normal)
Left, Right, Wide Center, Surround decoding **Modes[]** = 0907_{16}	(Dolby Pro Logic Wide)

The *predefined_ChannelConfig* field of the *cluster_information* of the function block must at least contain the union of all bits set for all the supported modes.

The following table outlines the combination of the common and process-specific *processing function block (Dolby Pro Logic)* descriptors. It is identical to the *processing function block descriptor (Up/Down-mix)* except for some field values. It is repeated here for clarity.

Table 8.8 – Dolby Pro Logic Processing function dependent information

Offset	contents	description
00 00_{16}	process_type (DOLBY_PRO_LOGIC)	identifier of sub-type processing.
00 01_{16}	size_of_controls	Size in bytes of the Controls field: n
00 02_{16}		
00 03_{16} … 00 03_{16}+n-1	Controls	A bit set to 1 indicates that the mentioned control is supported: Bit 0: Enable Processing Bit 1: Mode Select Bit 2..(n*8-1): Reserved for Processing specific
00 03_{16}+n	number_of_modes	Number of modes supported by this Processing function block: m
00 04_{16} +n	Size of Modes (2)	Defines the size of the Modes Field in bytes = 2 for Dolby Pro-Logic processing function block
00 05_{16} + n	Modes[0]	Describes the active logical channels in mode 0.
00 06_{16} + n		
:	:	:
:		
00 05_{16}+n+((m-1)*2)	Modes[m-1]	Describes the active logical channels in mode m-1.
00 06_{16}+n+((m-1)*2)		

8.4.5 3D-Stereo Extender Processing Dependent Information

The *process_type* field of the common *processing function block descriptor* contains the value 3D-STEREO_EXTENDER_PROCESS.

The *processing function block (3D-Stereo extender)* has a single input fb-plug. Therefore, the *number_of_inputs* field shall contain the value 1.

Bit 0 of the *Controls* field represents the Enable Processing Control. Bit 1 indicates the availability of the Spaciousness Control.

The input cluster may contain a number of logical channels. The processing function block (3D-Stereo Extender) uses at least Left and Right logical input channels as input for the extension process. Obviously, these two channels must be present in the input cluster for the function block to operate properly.

The output cluster may contain logical channels other than Left and Right (these must be present) to facilitate connectivity within the audio function. Channels that are present in the output cluster but not in the input cluster must be muted. Channels other than Left and Right that are present in both input and output cluster can be passed unaltered from input to output. Channels only present in the input cluster are absorbed by the *processing function block.*

There is no process-specific descriptor for the *3D-Stereo extender processing function block.*

Table 8.9 – processing function_block_type_dependent_information (3D-stereo extender)

address offset	contents	description
00 00_{16}	process_type (3D-stereo extender)	identifier of sub-type of processing
00 01_{16}	size_of_controls	Size in bytes of the Controls field: n
00 02_{16}		
00 03_{16}	Controls	A bit set to 1 indicates that the mentioned control is supported: Bit 0: Enable Processing Bit 1: Spaciousness Control Bit 2..(n*8-1): reserved for processing specific
:		
00 03_{16}+n-1		

8.4.6 Reverberation Processing Dependent Information

The *process_type* field of the common *processing function block descriptor* contains the value REVERBERATION_PROCESS.

The *reverberation processing function block* has a single input fb-plug. Therefore, the *number_of_inputs* field shall contain the value 1.

The *Controls* field indicates which reverberation-related Controls are effectively implemented in the *reverberation processing function block.*

The following table details the encoding of the Controls field for the *reverberation processing function block.* A bit set to 1 indicates that the mentioned control is supported.

Table 8.10 – Encoding of the control field for the reveration processing function block

Bit	corresponds to
0	Enable Processing.
1	Reverberation Type.
2	Reverberation Level.
3	Reverberation Time.
4	Reverberation Delay Feedback.
5	Reverberation Early Time
6..(n*8-1)	Reserved

Table 8.11 – Reverberation processing block dependent information

address offset	contents	description
00 00_{16}	process_type	identifier of sub-type of processing
00 01_{16}	size_of_controls	Size in bytes of the Controls field: n
00 02_{16}		
00 03_{16} … 00 03_{16}+n-1	Controls	A bit set to 1 indicates that the mentioned control is supported.

8.4.7 Chorus Processing Dependent Information

The *process_type* field of the common processing function block descriptor contains the value CHORUS_PROCESS.

The *chorus processing function block* has a single input fb-plug. Therefore, the *number_of_inputs* field shall contain the value 1.

The *Controls* field indicates which chorus-related Controls are effectively implemented in the *chorus processing function block*.

The *chorus processing function block descriptor* is identical to the common *processing function block descriptor*, except for some field values.

Table 8.12 – Encoding of the control field for the chorus processing function block

Bit	corresponds to
0	Enable Processing.
1	Chorus Level.
2	Chorus Modulation Rate.
3	Chorus Modulation Depth.
4..(n*8-1)	Reserved

Table 8.13 – Chorus processing block dependent information

address offset	contents	description
00 00_{16}	process_type	identifier of sub-type of processing
00 01_{16}	size_of_controls	Size in bytes of the Controls field: n
00 02_{16}		
00 03_{16} … 00 03_{16}+n-1	Controls	A bit set to 1 indicates that the mentioned control is supported:

8.4.8 Dynamic Range Compressor Processing Dependent Information

The *process_type* field of the common *processing function block descriptor* contains the value DYN_RANGE_COMP_PROCESS.

The *dynamic range compressor processing function block* has a single input fb-plug. Therefore, the *number_of_inputs* field shall contain the value 1.

The *Controls* field indicates which Controls are effectively implemented in the *dynamic range compressor processing function block.*

The *dynamic range compressor processing function block* descriptor is identical to the common processing function block descriptor, except for some field values.

Table 8.14 – Compression processing block dependent information

address offset	contents	description
00 00_{16}	process_type	identifier of sub-type of processing
00 01_{16}	size_of_controls	Size in bytes of the Controls field: n
00 02_{16}		
00 03_{16} … 00 03_{16}+n-1	Controls	A bit set to 1 indicates that the mentioned control is supported:

Table 8.15 – Encoding of the control field for the compression processing function block

Bit	corresponds to
0	Enable Processing.
1	Compression Ratio.
2	MaxAmpl.
3	Threshold.
4	Attack time.
5	Release time.
6..(n*8-1)	Reserved

8.5 CODEC Function Block Dependent Information

The *CODEC function block* is uniquely identified by the values in the function_block_type and the function_block_ID fields of the common function block dependent information.

The *source_ID* field is used to describe the connectivity for this CODEC function block. It contains the ID and type of the function block to which this CODEC function block is connection via its fb-plug.

The *Controls* field is a bitmap, indicating the availability of certain audio controls in the CODEC function block. For future expandability, the number of bytes occupied by the Controls field is indicated in the *size_of_controls* field.

The connectivity of the input fb-plug is described via the *source_ID* field. The *cluster_information*, describing the logical channels entering the CODEC function block are not repeated here. It is up to the controller software to trace the connections ‘upstream’ in order to locate the *cluster_information* pertaining to the audio channel cluster.

Table 8.16 – CODEC function block fb-plug and control information

address offset	contents	description
00 00_{16}	CODEC_type	identifier of sub-type of CODEC
00 01_{16}	size_of_controls (n)	Size in bytes of the Controls field
00 02_{16}		
00 03_{16}	Controls	A bit set to 1 indicates that the mentioned control is supported: Bit 0: Enable Processing Bit 1: Mode Select Bit 2..(n*8-1): Reserved for specific CODEC-types
:		
00 03_{16}+n	number_of_modes	Number of modes, supported by this processing function block: m.
00 04_{16}+n	size_of_modes (2)	Defines the size of the Modes Field in bytes = 2 for CODEC function block
00 05_{16}+n	Modes[0]	Describes the active logical channels in mode 0.
00 06_{16}+n		
:	:	:
:		
00 05_{16}+n+((m-1)*2)	Modes[m-1]	Describes the active logical channels in mode m-1.
00 06_{16}+n+((m-1)*2)		
	Format Specific	

8.5.1 DTS Decoder Dependent Information

Regardless of what kind of post-processing is chosen to add to DTS Digital Surround™, the only requirement DTS mandates is that there be a “bypass” or “DTS-only” mode in the system. In other words, DTS Digital Surround™ must exist independently as a discrete 5.1 channel decoder. The “bypass” or “DTS-only” mode for DTS Digital Surround™ must be implemented similarly to the concept of an ON/OFF switch (the enable processing control). As result, the 5.1 channel outputs should be level-matched with consistent gain. Furthermore, there is the need to maintain minimum compatibility to stereo systems. Hence, DTS mandates a 2/0 or Lt/Rt downmix mode in addition to the full 5.1 channel decoded output capability to be included in all DTS systems (except for A/V Decoders without any control features) that are designed with a DTS Digital Surround™ decoder IC.

The *CODEC_type* field of the common *CODEC_function_block* dependent information contains the value DTS_PROCESS.

The *CODEC function block* has a single input fb-plug. Therefore, the *number_of_inputs* field shall contain the value 1. The output *cluster_information* of the *CODEC function block* describes which logical channels are physically present at the output fb-plug of the *CODEC function block*. Depending on the selected operating mode, one or more channels may be unused.

Bit 0 of the *Controls* field represents the Enable Processing Control. The Mode Select Control (Bit 1) is used to change the behavior of the *CODEC function block* by selecting different modes of operation.

The process-specific descriptor of the *CODEC function block* describes the supported modes of operation. Selecting a mode of operation is done by manipulating the Mode control. The number of supported modes (*m*) is contained in the *number_of_modes* field. This field is followed by an array of mode fields, *Modes[]*. The index *I* into this array is zero-based and directly related to the number of the mode described by entry *Modes[I]*. It is the value *I* that must be used as a parameter for the audio command to select or retrieve the mode *I*.

The bit allocations in the *Modes[]* fields are very similar to those of the *predefined_ChannelConfig* field in a *cluster_information* i.e. a bit set in the *Modes[I]* field indicates that for mode *I*, the *CODEC function block* produces meaningful audio data for the logical channel that is associated with the position of the set bit. Logical channels that are present in the output cluster but are not used in a certain mode are considered to be inactive and at most produce silence in that mode.

Each *Modes[I]* field can only contain set bits for those logical channels that are present in the output channel cluster. In other words, all *Modes[]* fields can only contain a subset of the *predefined_ChannelConfig* field of the cluster_information of the function block. Furthermore, logical channels that have a non-predefined spatial position can not be marked as active in the *Modes[]* fields. It is therefore assumed by default that they are active.

The *size_of_controls* field equals 1.

Table 8.17 – DTS Decoder Dependent Data

Offset	contents	description
$00\ 00_{16}$	DTSCapabilities	Bitmap identifying the DTS capabilities of the decoder. A bit set to 1 indicates that the capability is supported: D0: Down-mixing 5.1 to matrix'ed stereo D1: Additional 5.1 down-mixing D2: Dynamic Range Control D3: Differential Time Delay D4..7: Reserved.
$00\ 01_{16}$	DTSFeatures	Bitmap identifying the features the decoder supports. A bit set indicates that the feature is supported: D0…7: Reserved.

Table 8.18 – Encoding of the control field for the DTS CODEC function block

Bit	corresponds to
0	Enable Processing.
1	Mode_Select
2	LFE_Select
3	Matrixed_Stereo_Select
4..(n*8-1)	Reserved

8.5.2 MPEG Decoder Dependent Information

An MPEG CODEC does not process MPEG transport streams directly. The subunit plug could create a program stream from an MPEG transport stream.

The *MPEGCapabilities* bitmap field describes the capabilities of the MPEG decoder.

Some general information must be retrieved from the Format Type-specific descriptor. For instance, the sampling frequencies supported by the decoder are reported through the Format Type-specific descriptor. This includes the ability of the decoder to handle low sampling frequencies (16 kHz, 22.05 kHz and 24 kHz) besides the standard 32 kHz, 44.1 kHz and 48 kHz sampling frequencies.

Bits D2..0 of the *MPEGCapabilities* field are used to indicate which layers this decoder is capable of processing. The different layers relate to the different algorithms that are used during encoding and decoding.

Bit D3 indicates that the decoder can only process the MPEG-1 base stream. Therefore, only Left and Right channels will be output.

Bit D4 indicates that the decoder supports the MPEG dual channel mode. In this case, the MPEG-1 base stream does not contain Left and Right channels of a stereo pair but instead contains two independent mono channels. One of these channels can be selected FUNCTION BLOCK command (Dual Channel Control) and reproduced over the Left and Right output channels simultaneously.

Bit D5 indicates that the decoder can handle MPEG-2 streams that contain two independent stereo pairs instead of the normal 3/2 encoding scheme. This bit is only applicable for MPEG-2 decoders.

Bit D6 indicates that the decoder supports the DVD MPEG-2 augmentation to 7.1 channels instead of the standard 5.1 channels.

Bit D7 indicates that the decoder is capable of processing streams that are encoded using adaptive multi-channel prediction.

Bits D9..8 indicate if the decoder can process embedded multilingual information. Multilingual capabilities can consist of being able to process multilingual information encoded at the same sampling frequency as the main audio channels (D9..8 = ‘01’). Some decoders may provide the additional capability to process multilingual information encoded at half the sampling frequency of the main audio channels (D9..8 = ‘11’).

D10 indicates that the decoder can support Main profile of MPEG2 AAC.

D11 indicates that the decoder can support Low Complexity (LC) profile of MPEG2 AAC.

D12 indicates that the decoder can support Scaleable Sampling Rate (SSR) profile of MPEG2 AAC. For more detail of each profile, please see ISO/IEC 13818-7 clause 7.

D13 indicates that the decoder can process Audio Data Interchange Format (ADIF) bit stream of MPEG2 AAC.

D14 indicates that the decoder can process Audio Data Transport Stream (ADTS) of MPEG2 AAC. For more detail of each stream, please see ISO/IEC 13818-7 clause 6 and 8.

D15 is reserved for future extension.

The *MPEGFeatures* field indicates compression-related features.

Bits D5..4 report which type of Dynamic Range Control the MPEG decoder supports. Some decoders do not implement DRC (D5..4 = '00'). If implemented, the DRC can either use the stream embedded gain parameters as is (D5..4 = '01') or can provide for additional DRC scaling factors, either a single scaling factor that influences both the boost and cut value simultaneously (D5..4 = '10') or a separate scaling factor for the boost and the cut value (D5..4 = '11')

All other bits are reserved.

Table 8.19 – MPEG Decoder Dependent Data

address offset	contents	description
00 00_{16}	MPEGCapabilities	Bitmap identifying the MPEG capabilities of the decoder. A bit set indicates that the capability is supported: D2..0: Layer support: D0 = Layer I D1 = Layer II D2 = Layer III D3: MPEG-1 only. D4: MPEG-1 dual-channel. D5: MPEG-2 second stereo. D6: MPEG-2 7.1 channel augmentation. D7: Adaptive multi-channel prediction. D9..8: MPEG-2 multilingual support: 00 = Not supported 01 = Supported at Fs 10 = Reserved 11 = Supported at F_s and ½F_s. D10: MPEG-2 AAC Main profile support D11: MPEG-2 AAC LC profile support D12: MPEG-2 AAC SSR profile support D13: MPEG-2 AAC ADIF bit stream support D14: MPEG-2 AAC ADTS support D15: Reserved.
00 02_{16}	MPEGFeatures	Bitmap identifying the features the decoder supports. A bit set indicates that the feature is supported: D3..0: Reserved. D5..4: Internal Dynamic Range Control: 00 = not supported. 01 = supported but not scalable. 10 = scalable, common boost and cut scaling value. 11 = scalable, separate boost and cut scaling value. D7..6: Reserved.

Table 8.20 – Encoding of the control field for the MPEG CODEC function block

Bit	corresponds to
0	Enable Processing.
1	Mode_Select
2..(n*8-1)	Reserved

8.5.3 AC-3 Decoder Dependent Information

An AC-3 decoder must be capable to interpret bit-streams encoded using the syntax defined in the standard described in ref. [R18]. The information specific to a decoder capable of decoding AC-3 bit-streams is shown below.

Table 8.21 – AC-3 decoder dependent information

address offset	contents	description
00 00_{16}	BSID	The value in this field indicates that that BSID mode and all modes with a lesser value are supported.
00 01_{16}	AC3Features	A bit set to 1 indicates that the mentioned feature is supported: D0: RF mode D1: Line mode D2: Custom0 mode D3: Custom1 mode D5..4: Internal Dynamic Range Control: 00 =not supported. 01 =supported but not scalable. 10 =scalable, common boost and cut scaling value. 11 =scalable, separate boost and cut scaling value. D7..6: Dual Mono Control 00= Channel 1&2 as stereo 01= Channel 1 as mono 10= Channel 2 as mono 11= Channel 1&2 mixed as mono D8: Dolby Surround Ex D15-9: reserved

The *BSID* field describes which bit stream identification this decoder is capable of processing. BSID modes can range from 0 to 31. The value in this field indicates that that BSID mode and all modes with a lesser value are supported. Standard AC-3 decoders must be capable of processing at least BSID modes 0 to 8 (meaning they are capable of interpreting the version 8 of the AC-3 bit-stream syntax and subsets).

The *AC3Features* bitmap field indicates compression-related features.

Bits D3..0 indicate which mode the decoder supports. To ease the design of decoder products, Dolby Digital ICs offer standard operating modes called “Line Mode” and “RF Mode.” These modes are included within the Dolby Digital decoder IC itself, thus greatly simplifying the implementation of dialog normalization, dynamic range control and down-mixing functions, all of which are necessary in Dolby Digital products. Two “Custom Modes” offer additional design flexibility aimed at more esoteric audio products for which additional implementation cost and complexity are not of primary concern.

Bits D5..4. indicate which type of Dynamic Range Control the AC-3 decoder supports. Some decoders do not implement DRC (D5..4 = ‘00’). If implemented, the DRC can either use the stream embedded gain parameters as is (D5..4 = ‘01’) or can provide for additional DRC scaling factors: either a single scaling factor that influences both the boost and cut value simultaneously (D5..4 = ‘10’), or a separate scaling factor for the boost and the cut value (D5..4 = ‘11’)

Bits D7..6 report what type of control over AC-3 dual mono bitstreams the decoder supports. In the case of dual mono data (also known as 1+1 input channels), the AC-3 stream does not contain Left and Right channels of a stereo pair but instead contains two independent mono channels. All AC-3 decoders must decode 1+1 mode bitstreams into linear PCM, but the decoder has some flexibility in determining how these two channels will be placed into the outgoing PCM bitstream using FUNCTION BLOCK command Dual Mono Control. The two channels can be delivered as Left and Right stereo, even though they contain independent data (D7..6 = '00'). The incoming Channel One can be delivered to both Left and Right PCM channels (D7..6 = 01). The incoming Channel Two can be delivered to both Left and Right PCM channels (D7..6 = 10). The final choice is that the two channels can be mixed together and the resulting summation written to both Left and Right PCM channels.

Table 8.22 – Encoding of the control field for the AC-3 CODEC function block

Bit	corresponds to
0	Enable Processing.
1	Mode_Select
2..(n*8-1)	Reserved

9. Function Block Command

FUNCTION BLOCK commands are always directed to a single function block within the subunit. The command contains enough information (*function_block_ID*, function_*block_type*) for the subunit to decide to where a specific command must be routed.

	msb							lsb
Opcode	FUNCTION BLOCK command ($B8_{16}$)							
Operand[0]	function_block_type							
Operand[1]	function_block_ID							
Operand[2]	control_attribute							
Operand[3]	function block specific							
:								
:								

Figure 9.1 – General format of the FUNCTION BLOCK command

A single opcode is used to indicate the command is addressed to a function block within the subunit. The value of opcode is $B8_{16}$. (Common unit and subunit command)

9.1 Function Block Command Components

To distinguish between actions to be taken when receiving the command, the ctype field should be decoded. For the **ctype** field, the value of “CONTROL” is used to “set the value of the control” and the value of “STATUS” is used to “get the value of the control”. The “NOTIFY” value is used to indicate notification is required when the control changes state.

9.1.1 Opcode

The value of the FUNCTION BLOCK command **opcode** is $B8_{16}$. (Common Unit and Subunit command).

9.1.2 Function_block_type

The *function_block_type* field (operand[0]) specifies the type of the function block (e.g., Selector, Feature, Processing,…). Table 9.1 – Function block type encoding shows the general encoding spaces for function block types for different subunit types. The type value is used to indicate the type of function block and subunit plug type in source_ID data structures.

Table 9.1 – Function block type encoding

Function block type	Meaning
$00\text{-}7F_{16}$	reserved
$80\text{-}8F_{16}$	audio subunit dependent
$90\text{-}EF_{16}$	reserved
$F0_{16}$	subunit destination plug
$F1_{16}$	subunit source plug
$F2_{16}\text{-}FD_{16}$	Reserved
FE_{16}	not connected
FF_{16}	Reserved for extension purposes

The *function_block_type* field (Operand[0]) is used to identify the type of function block and indicate the connection for the function block plug or subunit plug. The values 00_{16} – $7F_{16}$ are reserved to indicate the type of the common function block. The values 80_{16} – EF_{16} are used to indicate the type of the function block depending on the subunit. The values 80_{16} – $8F_{16}$ indicate the type of the function block depending on the audio subunit. The values 90_{16} – EF_{16} are reserved.

The values $F0_{16}$ – FE_{16} are used to indicate the particular information. For example, the values $F0_{16}$ and $F1_{16}$ are used to indicate the connection between the function block plug and subunit plug. And the value FE_{16} is used to indicate that a certain plug is not connected.

9.1.3 Function_block_ID

The *function_block_ID* field (operand[1]) specifies the identifier of the function block. The ID starts counting from 01_{16} to FE_{16}. The number 00_{16} is reserved. FF_{16} is reserved for extension purposes.

9.1.4 Control_attribute

The ***control_attribute*** field (operand[2]) specifies what kind of action will be performed on a control of a function block and which attribute of the control is addressed. In conjunction with the *ctype* field the operation to be performed is thus defined completely.

Usually, the control_attributes RESOLUTION, DELTA, MINIMUM, MAXIMUM, DEFAULT and DURATION are only used to get static settings with *ctype* = STATUS. On the other hand, the control_attributes CURRENT, MOVE and DELTA are used to set instances dynamically with ctype = CONTROL.

In particular, the control_attribute CURRENT specifies absolute settings for controls, while MOVE and DELTA specify relative settings for controls. Also, CURRENT is used to get the setting of a control with ctype = STATUS or ctype = NOTIFY.

The control attributes RESOLUTION, MINIMUM and MAXIMUM provide an actual scale of a control. RESOLUTION defines a minimum scale resolution of the control and is always positive. MINIMUM and MAXIMUM define the upper boundary and lower boundary of the control respectively. If a command requests a change resulting in a value outside of the scale, the command will be rejected. Within the scale, if a requested value in a command is not just on the scale resolution, the target device will accept the command and round off the value to the nearest realizable setting.

DEFAULT defines the default setting for a control.

DURATION defines the minimum time period in which a target device keeps on changing a control. When subsequently issuing a command with control_attribute = MOVE addressing the same control, the control will change for a maximum time equal to STEPS times the value of its DURATION attribute.

CURRENT defines the current setting of a control. When this control_attribute is used with ctype = CONTROL, the command requests a change of the addressed control to a setting specified in the dependent data field. The setting is an absolute value of a control. When this control_attribute is used with ctype = STATUS, the command requests to return a current setting of the corresponding control.

MOVE is used to request a change in a control. This control attribute is used only with ctype = CONTROL. A value of a control is not specified directly, but instead a count of DURATION is specified.

Using the MOVE control_attribute, smooth and continuous control can be realized by sending commands repeatedly. Then, a controller determines the count of DURATION and repetition interval by considering the value of DURATION and response time of a target. This mechanism is developed so a controller can make a target keep on changing a physical parameter without having trouble of overrunning. Namely, overrun caused by a communication error like a bus reset should be avoided. Therefore, controllers are expected to set the count of DURATION to a small number and to repeat sending commands in as short intervals as possible. Also, targets are not expected to support high counts of DURATION. This is informative, but 500ms for the total duration is sufficient for local bus applications. A longer total duration may be required for bus reset proof or bridged environments, but it is not in the scope of this general specification.

DELTA defines a request for a change in a control. This control attribute is used only with ctype = CONTROL. The value of the control is not specified directly, but instead, a count of RESOLUTION is specified.

Table 9.3 – Control_Attribute encoding defines the available control attributes.

Table 9.2 – Ctype and Control_attribute illustrates what values of ctype are recommended (R) or optional (O) for which control_attributes. A dash in the Supported ctype column indicates that the Control_Attribute is not supported for the defined *ctype*

Table 9.2 – Ctype and Control_attribute

Control_Attribute	Supported ctype			Comments
	C	S	N	
RESOLUTION	O	R	O	Minimum scale of a control
MINIMUM	O	R	O	Minimum setting of a control
MAXIMUM	O	R	O	Maximum setting of a control
DEFAULT	O	R	O	Default setting of a control
DURATION	O	R	O	Minimum moving time of a control
CURRENT	R	R	O	Current setting of a control
MOVE	R	–	–	Request to change a value during a period equal to a number of DURATION
DELTA	R	-	-	Relative setting of a control in unit steps

The control_attribute support depends on function block and control. For example, mute control in Feature function block can support only control_attribute CURRENT. On the other hand, volume control can support CURRENT, MINIMUM, MAXIMUM, DEFAULT, DURATION, MOVE, DELTA, and RESOLUTION. Which control_attribute each function block support is described in the following sub clauses.

Also the *control_attribute* field of FUNCTION BLOCK command addressed to each function block depends on implementation. For example, volume control in one product supports eight control_attributes, CURRENT, MINIMUM, MAXIMUM, DEFAULT, DURATION, MOVE, DELTA, and RESOLUTION, but another product supports only CURRENT. It is also possible that volume control in one product supports MOVE, DELTA with *ctype* = CONTRL, and CURRENT only with *ctype* = STATUS.

Table 9.3 – Control_Attribute encoding

Value	Subfunction
00_{16}	reserved
01_{16}	RESOLUTION
02_{16}	MINIMUM
03_{16}	MAXIMUM
04_{16}	DEFAULT
05_{16}-07_{16}	reserved
08_{16}	DURATION
09_{16}-$0F_{16}$	reserved
10_{16}	CURRENT
11_{16}-17_{16}	reserved
18_{16}	MOVE
19_{16}	DELTA
$1A_{16}$-FF_{16}	reserved

10. Audio Subunit FUNCTION_BLOCK command

This section defines commands that are applicable to an audio subunit.

The FUNCTION BLOCK command is used to modify/retrieve the values of audio related controls. A single opcode used to indicate the command is addressed to a function block within the audio subunit.

Control of function blocks within an audio subunit is performed through the manipulation of individual controls, embedded in the function blocks of the audio subunit. The *function_block_dependent_information* indicates which controls are present in a particular function block.

FUNCTION BLOCK commands are always directed to a single function block within the subunit. The command contains enough information (function block ID, function block type, fb-plugID, *audio channel number*(s) and Control Selector) for the subunit to decide to where a specific command must be routed.

The *subunit_type* and *subunit_ID* fields define the address of the recipient of the command or the source of the response as defined in the General Specification of AV/C Digital Interface Command Set. Audio subunit type is 01_{16}.

	msb							lsb
	0000							ctype
	subunit_type (audio subunit = 01_{16})							subunit_ID
Opcode	FUNCTION BLOCK command ($B8_{16}$)							
Operand[0]	function_block_type							
Operand[1]	function_block_ID							
Operand[2]	control_attribute							
Operand[3]	selector_length (n+1)							
Operand[3+1]	audio_selector_data[1]							
…	…							
Operand[3+n]	audio_selector_data[n]							
Operand[3+n+1]	control_selector							
Operand[4+n+1]	control_data (x)							
…								
Operand[4+n+x]								

Figure 10.1 – Format of the audio subunit FUNCTION BLOCK command

This command format is defined for the audio subunit type or its derivatives.

The value of the *selector_length* field includes the *audio_selector_data* and the *control_selector*. It does not include itself.

audio_selector_data[] field contains information for audio signal selection such as input FB-plug number, input or output audio_channel_number. The detail of this field is specified in each function block.

control_selector field indicates which type of control this command is manipulating.

control_data field has *control_selector* specific data. The detail of this field is described in the following sub clauses.

10.1 FUNCTION BLOCK Command Components

To distinguish between actions to be taken when receiving the command, the ctype field should be decoded. For the *ctype* field, the value of “CONTROL” is used to “set the value of the control” and the

value of “STATUS” is used to “get the value of the control”. By the same token the “NOTIFY” value is used to indicate notification is required when the control changes state.

10.1.1 Function_block_type

The ***function_block_type*** field (operand[0]) specifies the type of the function block (e.g., Selector, Feature, Processing,…). Table 10.1 – Function block type encoding shows the general encoding spaces for function block types for different subunit types. The type value is used to indicate the type of function block and subunit plug type in source_ID data structures.

Table 10.1 – Function block type encoding

Function block type	Meaning
$00\text{-}7F_{16}$	reserved
$80\text{-}8F_{16}$	audio subunit dependent
$90\text{-}EF_{16}$	reserved
$F0_{16}$	subunit destination plug
$F1_{16}$	subunit source plug
$F2_{16}\text{-}FD_{16}$	Reserved
FE_{16}	not connected
FF_{16}	Extension

Table 10.2 – Audio Function block type encoding shows the coding for the different function block types as defined for the audio subunit.

Table 10.2 – Audio Function block type encoding

Function block type	Meaning
80_{16}	Selector function block
81_{16}	Feature function block
82_{16}	Processing function block
83_{16}	CODEC function block
$84_{16}\text{-}8F_{16}$	Reserved for future use

10.2 Selector function block

	msb							lsb
Operand[3]	selector_length (2)							
Operand [4]	Input fb-plug number (FBPN)							
Operand[5]	control_selector (SELECTOR_CONTROL)							

Figure 10.2 – Function block command for Selector function block

audio_selector_data[] fields for selector FUNCTION BLOCK command consists of *Input_fb_plug_number* fields. There are no *control_data* fields for selector FUNCTION BLOCK command.

This command is used to modify/retrieve the Selector Control inside a Selector Function block of the audio function.

Possible control_attributes supported by a selector function block are:

— CURRENT

— MINIMUM

— MAXIMUM

— DEFAULT

The operand[1] field specifies the *selector function_block*_ID. The values in operand[1] must be appropriate to the audio subunit. Only existing *selector function blocks* in the audio function can be addressed. If the command specifies an unknown or non-existing SFBID, the audio subunit shall return an AV/C response frame with response code "NOT IMPLEMENTED".

An audio subunit that receives this command with *ctype* value of CONTROL should connect the output fb-plug to the selected input fb-plug. The input fb-plug is indicated in Input fb-plug number field (operand[4]). The *Input fb-plug number* field (operand[4]) in the control command shall not be set to FF_{16}

On the other hand, an audio subunit that receives this command with *ctype* value of STATUS should return the current setting of the Selector Control inside a *selector function block* of the audio subunit.

In this case, *Input fb-plug number* field (operand[4]) is set to FF_{16} when the status command is issued, and is updated to the current setting of the *control_attribute* when the STABLE response frame is returned.

control_selector field(operand[5]) should have value SELECTOR_CONTROL(01_{16}).

When the setting of a selector function block is requested, the actual parameter for the selector control is returned in the response frame in operand[4].

The FUNCTION BLOCK command with a *control_selector* of SELECTOR_CONTROL may be used as status command. This command has a *ctype* value of STATUS. In this case, *operand[4]* is set to FF_{16} when the status command is issued and is updated to the current setting of the *control_attribute* when the STABLE response frame is returned.

The operand[4] in the control command shall not be set to FF_{16}.

10.3 Feature function block

The following paragraphs describe the FUNCTION BLOCK commands for Feature function block These commands can have two forms. The first form must be supported while the second form can be optionally implemented.

	msb							lsb
Operand[3]	2							
Operand[4]	audio channel number (ICN)							
Operand[5]	control_selector							
Operand[5+1] … Operand[5+n]	control_parameters							

Figure 10.3 – Feature function block command

The function block command for feature function block is used to set/get/notify the settings of the attributes of an audio control inside a particular *feature function block* of the audio function.

The *control_attribute* field of the command contains a constant, identifying which attribute of the addressed control is to be modified/retrieved. Possible attributes for a control for a feature function block are:

- CURRENT
- MINIMUM
- MAXIMUM
- RESOLUTION
- DEFAULT
- DURATION
- DELTA
- MOVE

If the addressed Control does not support modification/readout of a certain attribute, the response frame shall indicate NOT IMPLEMENTED when an attempt is made to modify that attribute. In most cases, only the CURRENT attribute will be supported for the FUNCTION BLOCK command for a feature function block. However, this specification does not prevent a designer from making other attributes programmable.

The *selector_length* field (Operand[3]) for feature function block shall always be set to 2. The *audio_selector_data* shall consist of *audio_channel_number (ICN)* (Operand[4]) and *control_selector* (Operand[5]).

The operand[4] field specifies the *audio_channel_number (ICN)* of the control that the FUNCTION BLOCK command is addressed to. If the value of this field is 0, then the command is addressed to the "master" control. If the value is 255, the command is addressed to all controls of the type *control_selector* in the feature function block. If an unsupported or unimplemented control on the *audio_channel_number* is addressed, the response frame shall return NOT IMPLEMENTED. The value 254 indicates the channel number is "VOID" for this control_selector in the feature function block.

The operand[5] field specifies the *control_selector*The Control Selector indicates which type of control this command is manipulating. (Volume, Mute, etc.). If the command specifies an unknown or unsupported control_selector, the response frame shall indicate NOT IMPLEMENTED when an attempt is made to modify that attribute.

A special case arises when the *audio_channel_number* is set to FF_{16}. Then a single FUNCTION BLOCK command can be used to set/retrieve an attribute of all available controls of a certain type (indicated by the *control_selector*) within the Function block. The number of parameters passed in the parameter block must exactly match the number of available controls in the function block. If this is not the case, the response frame shall indicate NOT IMPLEMENTED. The first parameter in the parameter block is assigned to the attribute of the first available control, i.e. the one with the lowest *audio_channel_number*. The above description is referred to as the second form of the Set command for feature function block.

The operand[1] field specifies the feature *function_block_ID*. Only existing feature function blocks in the audio function can be addressed. If the command specifies an unknown or non-existing Feature *function_block_ID*, the response frame shall indicate NOT IMPLEMENTED..

The actual parameter(s) for the feature function block control are passed in the rest of the command/response frame. The length of the parameter block is indicated in the operand[5+1] field of the command.

The following paragraphs present a detailed description of all possible controls a feature function block can incorporate. For each control, the layout of the parameter block together with the appropriate Control Selector is listed for all forms of the feature function block command ctype=CONTROL.

10.3.1 Mute Control

The Mute Control is one of the optional building blocks of a Feature function block. A Mute Control can have only the current setting attribute (CURRENT). The setting of *Mute_On* field can be either TRUE (70_{16}) or FALSE (60_{16}), TRUE meaning muted and FALSE meaning not-muted. Also the value FF_{16} indicates invalidity and shall not be used for a control command.

In the first form of the function block command for feature function block, a particular Mute Control within a Feature function block is addressed through the Function block ID and *audio_channel_number* fields of the FUNCTION BLOCK command. The valid range for the *audio_channel_number* field is from zero (the 'master' channel) up to the number of logical channels in the audio channel cluster.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (1)							
Operand[7]	Mute_On							

Figure 10.4 – First Form of the Mute Control Parameters

Operand[5]	MUTE_CONTROL
Operand[6]	1

Figure 10.5 – Values for the fields in the first form of the Mute Control Parameters

In the second form, the *audio_channel_number* field is set to FF_{16}. The parameter block contains a list of settings for the CURRENT attribute for all available Mute Controls in the Feature function block.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (n)							
Operand[6+1]	Mute_On[1]							
Operand[6+n]	Mute_On[n]							

Figure 10.6 – Second Form of the Mute Control Parameters

Operand[5]	MUTE_CONTROL
Operand[6]	Number of available Controls: NrAv

Figure 10.7 – Second Form of the Mute Control Parameter Block

10.3.2 Volume Control

The Volume Control is one of the optional building blocks of a *feature function block*. A Volume Control can support the following Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT, DURATION, MOVE, DELTA and RESOLUTION). The settings for the CURRENT, MIN and MAX attributes can range from 127.9922dB ($7FFE_{16}$) down to –127.9961 dB (8001_{16}) in steps of 1/256 dB or 0.00390625 dB (0001_{16}), Code 8000_{16} represents silence, i.e. -∞dB. The settings for the RESOLUTION attribute can only take positive values and range from 1/256 dB (0001_{16}) to +127.9922dB ($7FFE_{16}$).

The settings for the MOVE and DELTA attribute are a number indicating a number of steps. They can range from +32766 ($7FFE_{16}$) down to –32767 (8001_{16}) in steps of 1 (0001_{16}).

The settings for the DURATION attribute can only take positive values and range from 0 msec (0000_{16}) to +32766 msec ($7FFE_{16}$) in steps of 1 msec (0001_{16}).

The Volume Control honors the command to the best of its abilities. It may round the Volume attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (2)							
Operand[7]	Steps_High							
Operand[8]	Steps_Low							

Figure 10.8 – Relative Form of the Volume Control Parameters

Operand[5]	VOLUME_CONTROL
Operand[6]	2

Figure 10.9 – Values for the fields in the relative form of the Volume Control Parameters

Table 10.3 – Values for the Steps with the MOVE and DELTA attributes

Value	corresponds to
$7FFF_{16}$	invalid
$7FFE_{16}$	+32766
…	…
…	…
0002_{16}	+2
0001_{16}	+1
0000_{16}	0
$FFFF_{16}$	-1
$FFFE_{16}$	-2
…	…
…	…
8001_{16}	-32767
8000_{16}	Reserved for future

Table 10.4 – Values for the Steps with the DURATION attribute

Value	corresponds to
$7FFF_{16}$	invalid
$7FFE_{16}$	+32766 msec
…	…
…	…
0002_{16}	+2 msec
0001_{16}	+1 msec
0000_{16}	0 msec
$FFFF_{16}$	Reserved for future standardization
$FFFE_{16}$	
…	
…	
8001_{16}	
8000_{16}	

The relative form of the Volume Control is used with the DELTA, MOVE and DURATION attributes.

The DURATION attribute is used with STATUS ctype in a command, and a minimum moving time period will be reported in Steps operand of a response

The MOVE attribute, with CONTROL ctype, defines request for time period in which a target device keeps on changing volume, then the time period is calculated from the DURATION multiplied by the number specified in Steps operand. If a target keeps on changing volume, it will stop changing when the value of Steps is set to 0000_{16}.

For example, the MOVE attribute is used on the mechanical volume control which drives a motor.

In the first form of the FUNCTION BLOCK command, a particular Volume Control within a feature function block is addressed through the *function_block_ID* and *audio_channel_number* fields of the FUNCTION BLOCK command. The valid range for the *audio_channel_number* field is from zero (the 'master' channel) up to the number of logical channels in the audio channel cluster.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (2)							
Operand[7]	Volume_High							
Operand[8]	Volume_Low							

Figure 10.10 – First Form of the Volume Control Parameters

Operand[5]	VOLUME_CONTROL
Operand[6]	2

Figure 10.11 – Values for the fields in the first form of the Volume Control Parameters

Table 10.5 – Values for the volume settings

Value	corresponds to
$7FFF_{16}$	invalid
$7FFE_{16}$	127.9922dB
…	…
0100_{16}	1.0000dB
…	…
0002_{16}	0.0078dB
0001_{16}	0.0039dB
0000_{16}	0.0000dB
$FFFF_{16}$	-0.0039dB
$FFFE_{16}$	-0.0078dB
…	…
$FF00_{16}$	-1.0000 dB
…	…
8002_{16}	-127.9922dB
8001_{16}	-127.9961dB
8000_{16}	-∞dB

In the second form, the *audio_channel_number* field is set to FF_{16}. The parameter block contains a list of settings for an attribute of all available Volume Controls in the *feature function block*.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (2*n)							
Operand[6+1]	Volume_High(1)							
Operand[6+2]	Volume_Low(1)							
…	…							
…	…							
Operand[6+2*n-1]	Volume_High(n)							
Operand[6+2*n]	Volume_Low(n)							

Figure 10.12 – Second Form of the Volume Control Parameters

Operand[5]	VOLUME_CONTROL
Operand[6]	Number of available Controls: NrAv)*2

Figure 10.13 – Values for the fields in the second form of the Volume Control Parameters

10.3.3 LR Balance Control

The LR Balance Control is one of the optional building blocks of a *feature function block* for level balance control between left and right channel groups.

In case ConfigType = GENERIC_MULTI_SPEAKER, the left and right channel groups are defined as below.

The Left channel group is:

- Left Front (L)
- Left Surround (L_S)
- Left of Center (L_C)
- Side Left (S_L)
- Left Front Effect (FEL)

The Right channel group is:

- Right Front (R)
- Right of Center (R_C)
- Right Surround (R_S)
- Side Right (S_R)
- Right Front Effect (FER)

The followings don't belong to the both groups and aren't influenced by the LR Balance Control:

- Center Front (C)
- Low Frequency Enhancement (LFE)
- Surround (S)
- Top (T)
- Bottom (B)

The LR Balance Control is predicated on the relation between the left and right channel groups. A LR Balance Control bit of Controls[0] in Feature function block dependent data should be set to 1, and other LR Balance bits of Control[n] should be reset to 0. Similarly the *audio_channel_number* (ICN) in the control command should be zero. If a FUNCTION BLOCK command for LR Balance control contains an *audio_channel_number* other than zero, the response frame should return NOT IMPLEMENTED.

A LR Balance Control can support following Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT, DURATION, MOVE, RESOLUTION and DELTA). The settings for the CURRENT, MIN and MAX attributes can range from 0/-127.9922 dB ($7FFE_{16}$) through 0/0 dB (0000_{16}) to –127.9961/0 dB (8001_{16}) in steps of 1/256 dB or 0.00390625 dB (0001_{16}) at each reducing area, Code $7FFE_{16}$ and 8000_{16} represent silence for one of a pair, i.e. -∞dB. The settings for the RESOLUTION attribute can only take positive values and range from 1/256 dB (0001_{16}) to +127.9922 dB ($7FFE_{16}$).

The LR Balance Control honors the command to the best of its abilities. It may round the LR Balance attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

In the first form of the *FUNCTION BLOCK command*, a particular LR Balance Control within a *feature function block* is addressed through the function_block_ID and *audio_channel_number* fields of the FUNCTION BLOCK command. The valid range for the *audio_channel_number* field is zero (the 'master' channel).

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (2)							
Operand[7]	LR_Balance _High							
Operand[8]	LR_Balance _Low							

Figure 10.14 – Form of the LR Balance Control Parameters

Operand[5]	LR_BALANCE_CONTROL
Operand[6]	2

Figure 10.15 – Values for the fields in the first form of the LR Balance Control Parameters

Table 10.6 – Values for the LR Balance settings

Value	corresponds to	
	Left channel group	Right channel group
$7FFF_{16}$	invalid	invalid
$7FFE_{16}$	0.0000dB	-∞dB
$7FFD_{16}$	0.0000dB	-127.9883dB
…	0.0000dB	…
0100_{16}	0.0000dB	-1.0000dB
…	0.0000dB	…
0002_{16}	0.0000dB	-0.0078dB
0001_{16}	0.0000dB	-0.0039dB
0000_{16}	0.0000dB	0.0000dB
$FFFF_{16}$	-0.0039dB	0.0000dB
$FFFE_{16}$	-0.0078dB	0.0000dB
…	…	0.0000dB
$FF00_{16}$	-1.0000 dB	0.0000dB
…	…	0.0000dB
8002_{16}	-127.9922dB	0.0000dB
8001_{16}	-127.9961dB	0.0000dB
8000_{16}	-∞dB	0.0000dB

10.3.4 FR Balance Control

The FR Balance Control is one of the optional building blocks of a *feature function block* for level balance control between left and rear channel groups.

In case ConfigType = GENERIC_MULTI_SPEAKER, the front and rear channel groups are defined as below.

- The front channel group is:Left Front (L)
- Right Front (R)
- Center Front (C)
- Left of Center (L_C)
- Right of Center (R_C)
- Left Front Effect (FEL)
- Right Front Effect (FER)
- The Rear channel group is:Left Surround (L_S)
- Right Surround (R_S)
- Surround (S)

The followings don't belong to both groups and aren't influenced by the FR Balance Control

- Low Frequency Enhancement (LFE).
- Top (T)
- Bottom (B)
- The followings aren't defined:Side Left (S_L)
- Side Right (S_R)

The FR Balance Control is predicated on the relation between the front and rear channel groups. A FR Balance Control bit of Controls[0] in Feature function block dependent data should be set to 1, and other FR Balance bits of Control[n] should be reset to 0. Similarly the *audio_channel_number* (ICN) in the control command should be zero. If a FUNCTION BLOCK command for FR Balance control contains an *audio_channel_number* other than zero, the response frame should return NOT IMPLEMENTED. A FR Balance Control can support following Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT, DURATION, MOVE, RESOLUTION, and DELTA). The settings for the CURRENT, MIN and MAX attributes can range from 0/-127.9922 dB ($7FFE_{16}$) through 0/0 dB (0000_{16}) to –127.9961/0 dB (8001_{16}) in steps of 1/256 dB or 0.00390625 dB (0001_{16}) at each reducing area, Code $7FFE_{16}$ and 8000_{16} represent silence for one of a pair, i.e. -∞dB. The settings for the RESOLUTION attribute can only take positive values and range from 1/256 dB (0001_{16}) to +127.9922dB ($7FFE_{16}$).

The FR Balance Control honors the command to the best of its abilities. It may round the FR Balance attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

In the first form of the *FUNCTION BLOCK command*, a particular FR Balance Control within a *feature function block* is addressed through the function_block_ID and ,*audio_channel_number* fields of the FUNCTION BLOCK command.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (2)							
Operand[7]	FR_Balance _High							
Operand[8]	FR_Balance _Low							

Figure 10.16 – Form of the FR Balance Control Parameters

Operand[5]	FR_BALANCE_CONTROL
Operand[6]	2

Figure 10.17 – Values for the fields in the first form of the FR Balance Control Parameters

Table 10.7 – Values for the FR Balance settings

Value	corresponds to	
	Front channel group	Rear channel group
$7FFF_{16}$	invalid	invalid
$7FFE_{16}$	0.0000dB	-∞dB
$7FFD_{16}$	0.0000dB	-127.9883dB
…	0.0000dB	…
0100_{16}	0.0000dB	-1.0000dB
…	0.0000dB	…
0002_{16}	0.0000dB	-0.0078dB
0001_{16}	0.0000dB	-0.0039dB
0000_{16}	0.0000dB	0.0000dB
$FFFF_{16}$	-0.0039dB	0.0000dB
$FFFE_{16}$	-0.0078dB	0.0000dB
…	…	0.0000dB
$FF00_{16}$	-1.0000 dB	0.0000dB
…	…	0.0000dB
8002_{16}	-127.9922dB	0.0000dB
8001_{16}	-127.9961dB	0.0000dB
8000_{16}	-∞dB	0.0000dB

10.3.5 Bass Control

The Bass Control is one of the optional building blocks of a *feature function block*. The Bass Control influences the general Bass behavior of the *feature function block*. A Bass Control can support following Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The

settings for the CURRENT, MINIMUM, MAXIMUM and DEFAULT attributes can range from +31.50 dB ($7E_{16}$) down to –32.00 dB (80_{16}) in steps of 0.25 dB (01_{16}). The settings for the RESOLUTION attribute can only take positive values and range from 0.25 dB (01_{16}) to +31.50 dB ($7E_{16}$). The Bass Control honors the command to the best of its abilities. It may round the *Bass attribute value* to its closest available setting. It will report this setting when queried during a FUNCTION BLOCK command. Other parameters that also influence the behavior of the Bass Control, such as cut-off frequency, cannot be altered through this command.

In the first form of the *FUNCTION BLOCK command*, a particular Bass Control within a *feature function block* is addressed through the function_block_ID and *audio_channel_number* fields of the *feature function block command*. The valid range for the *audio_channel_number* field is from zero (the 'master' channel) up to the number of logical channels in the audio channel cluster.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (1)							
Operand[7]	Bass							

Figure 10.18 – First Form of the Bass Control Parameters

Operand[5]	BASS_CONTROL
Operand[6]	1

Figure 10.19 – Values for the fields in the first form of the Bass Control Parameters

Table 10.8 – Settings for the Bass Control attribute

Value	corresponds to
$7F_{16}$	invalid
$7E_{16}$	+31.50dB
…	…
00_{16}	0.00dB
…	…
82_{16}	-31.50dB
81_{16}	-31.75dB
80_{16}	-32.00 dB

In the second form, the *audio_channel_number* field is set to FF_{16}. The parameter block contains a list of settings for all available Bass Controls in the *feature function block*.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (n)							
Operand[6+1]	Bass[1]							
…	…							
Operand[6+n]	Bass[n]							

Figure 10.20 – Second Form of the Bass Control Parameters

Operand[5]	BASS_CONTROL
Operand[6]	Number of available Controls: NrAv

Figure 10.21 – Values for the fields in the second form of the Bass Control Parameters

10.3.6 Mid Control

The Mid Control is one of the optional building blocks of a *feature function block*. The Mid Control influences the general Mid behavior of the *feature function block*. A Mid Control can support following Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The settings for the CURRENT, MINIMUM, MAXIMUM and DEFAULT attributes can range from +31.50 dB ($7E_{16}$) down to –32.00 dB (80_{16}) in steps of 0.25 dB (01_{16}). The settings for the RESOLUTION attribute can only take positive values and range from 0.25 dB (01_{16}) to +31.50 dB ($7E_{16}$). The Mid Control honors the command to the best of its abilities. It may round the Mid attribute value to its closest available setting. It will report this setting when queried by a FUNCTION BLOCK command. Other parameters that also influence the behavior of the Mid Control, such as cut-off frequency, cannot be altered through this command.

In the first form of the *FUNCTION BLOCK command*, a particular Mid Control within a *feature function block* is addressed through the *function_block_ID* and *audio_channel_number* fields of the FUNCTION BLOCK command. The valid range for the *audio_channel_number* field is from zero (the ‘master’ channel) up to the number of logical channels in the audio channel cluster.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (1)							
Operand[7]	Mid							

Figure 10.22 – First Form of the Mid Control Parameters

Operand[5]	MID_CONTROL
Operand[6]	1

Figure 10.23 – Values for the fields in the first form of the Mid Control Parameters

Table 10.9 – Settings for the Mid Control attribute

Value	corresponds to
$7F_{16}$	invalid
$7E_{16}$	+31.50dB
…	…
00_{16}	0.00dB
…	…
82_{16}	-31.50dB
81_{16}	-31.75dB
80_{16}	-32.00 dB

In the second form, the *audio_channel_number* field is set to FF_{16}. The parameter block contains a list of settings for all available Mid Controls in the Feature function block.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (n)							
Operand[6+1]	Mid[1]							
…	…							
Operand[6+n]	Mid[n]							

Figure 10.24 – Second Form of the Mid Control Parameters

Operand[5]	MID_CONTROL
Operand[6]	Number of available Controls: NrAv

Figure 10.25 – Values for the fields in the second form of the Mid Control Parameters

10.3.7 Treble Control

The Treble Control is one of the optional building blocks of a *feature function block*. The Treble Control influences the general Treble behavior of the *feature function block*. A Treble Control can support following Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The settings for the CURRENT, MINIMUM, MAXIMUM and DEFAULT attributes can range from +31.50 dB ($7E_{16}$) down to –32.00 dB (80_{16}) in steps of 0.25 dB (01_{16}). The settings for the RESOLUTION attribute can only take positive values and range from 0.25 dB (01_{16}) to +31.50 dB ($7E_{16}$). The Treble Control honors the command to the best of its abilities. It may round the Treble attribute value to its closest available setting. It will report this setting when queried during a FUNCTION BLOCK command. Other parameters that also influence the behavior of the Treble Control, such as cut-off frequency, cannot be altered through this command.

In the first form of the FUNCTION BLOCK command, a particular Treble Control within a Feature function block is addressed through the Function block ID and *audio_channel_number* fields of the Set/Get Feature function block command. The valid range for the *audio_channel_number* field is from zero (the ‘master’ channel) up to the number of logical channels in the audio channel cluster.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (1)							
Operand[7]	Treble							

Figure 10.26 – First Form of the Treble Control Parameters

Operand[5]	TREBLE_CONTROL
Operand[6]	1

Figure 10.27 – Values for the fields in the first form of the Treble Control Parameters

Table 10.10 – Settings for the Treble Control attribute

Value	corresponds to
$7F_{16}$	invalid
$7E_{16}$	+31.50dB
…	…
00_{16}	0.00dB
…	…
82_{16}	-31.50dB
81_{16}	-31.75dB
80_{16}	-32.00 dB

In the second form, the *audio_channel_number* field is set to FF_{16}. The parameter block contains a list of settings for all available Treble Controls in the *feature function block*.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (n)							
Operand[6+1]	Treble[1]							
…	…							
Operand[6+n]	Treble[n]							

Figure 10.28 – Second Form of the Treble Control Parameters

Operand[5]	TREBLE_CONTROL
Operand[6]	Number of available Controls: NrAv

Figure 10.29 – Values for the fields in the second form of the Treble Control Parameters

10.3.8 Graphic Equalizer Control

The Graphic Equalizer Control is one of the optional building blocks of a *feature function block*. The specification provides for standard support of a third octave graphic equalizer. The bands are defined according to the ANSI S1.11-1986 standard. Bands are numbered from 14 (center frequency of 25 Hz) up to 43 (center frequency of 20,000 Hz), making a total of 30 possible bands. The following table lists the band numbers and their center frequencies.

Table 10.11 – Band Numbers and Center Frequencies per ANSI S1.11-1986 standard

Band Nr.	Center Freq.	Band Nr.	Center Freq.	Band Nr.	Center Freq.
14	25Hz	24*	250Hz	34	2500Hz
15*	31.5Hz	25	315Hz	35	3150Hz
16	40Hz	26	400Hz	36*	4000Hz
17	50Hz	27*	500Hz	37	5000Hz
18*	63Hz	28	630Hz	38	6300Hz
19	80Hz	29	800Hz	39*	8000Hz
20	100Hz	30*	1000Hz	40	10000Hz
21*	125Hz	31	1250Hz	41	12500Hz
22	160Hz	32	1600Hz	42*	16000Hz
23	200Hz	33*	2000Hz	43	20000Hz

* Note: Bands marked with an asterisk (*) are those present in an octave equalizer.

Table 10.12 – Extra Band Numbers and Center Frequencies for a sixth octave equalizer

Band Nr.	Center Freq.	Band Nr.	Center Freq.	Band Nr.	Center Freq.
1	18Hz				
2	20Hz				
3	22Hz	13	220Hz	23	2200Hz
4	28Hz	14	280Hz	24	2800Hz
5	35Hz	15	355Hz	25	3550Hz
6	44.5Hz	16	445Hz	26	4450Hz
7	56Hz	17	560Hz	27	5600Hz
8	70Hz	18	710Hz	28	7100Hz
9	89Hz	19	890Hz	29	8900Hz
10	110Hz	20	1100Hz	30	11000Hz
11	140Hz	21	1400Hz	31	14000Hz
12	180Hz	22	1800Hz	32	18000Hz

A *feature function block* that supports the Graphic Equalizer Control is not required to implement the full set of filters. A subset (for example, octave bands) may be implemented. In a FUNCTION BLOCK command with = STATUS, the BandsPresent and ExtraBandsPresent fields in the parameter block are bitmaps indicating which bands are effectively implemented and thus reported back in the returned parameter block. Consequently, the number of bits set in this field determines the total length of the returned parameter block. In a FUNCTION BLOCK command with = CONTROL, a bit set in the (x)BandsPresent fields indicates there is a new setting for that band in the parameter block that follows. The new values must be ordered in ascending order. If the number of bits set in the (x)BandsPresent fields does not match the number of parameters specified in the following block, an AV/C response frame “NOT IMPLEMENTED” shall be returned.

A Graphic Equalizer Control can support following Control attributes: CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA. However, if a certain attribute is supported, it must do so for all individual bands. The settings for the CURRENT, MINIMUM, MAXIMUM and DEFAULT attributes can range from +31.50dB ($7E_{16}$) down to –32.00 dB (80_{16}) in steps of 0.25 dB (01_{16}). The settings for the RESOLUTION attribute can only take positive values and range from 0.25 dB (01_{16}) to +31.50dB ($7E_{16}$). The Equalizer Control honors the command to the best of its abilities. It may round the Bandxx attribute values to their closest available settings. It will report these settings when queried with a FUNCTION BLOCK command.

Only the first form of the FUNCTION BLOCK command for a Graphic Equalizer Control is supported. A particular Graphic Equalizer Control within a Feature function block is addressed through the Function block ID and *audio_channel_number* fields of the FUNCTION BLOCK command. The valid range for the *audio_channel_number* field is from zero (the ‘master’ channel) up to the number of logical channels in the audio channel cluster.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (4+n)							
Operand[6+1]	BandsPresent1							
…	…							
Operand[6+4]	BandsPresent4							
Operand[6+5]	ExtraBandsPresent1							
…	…							
Operand[6+8)	ExtraBandsPresent4							
Operand[6+9)	Gain[Lowest]							
…	…							
Operand[6+8+n]	Gain[Highest]							

Figure 10.30 – First Form of the Graphic Equalizer Control Parameters

Operand[5]	GEQ_CONTROL
Operand[6]	8+(number of bits set in BandsPresent : NrBits)

Figure 10.31 – Values for the fields in the first form of the GEQ Control Parameters

Table 10.13 – Encoding of the BandPresent parameter

Bitnumber	Meaning
Bit 0	Band 14 is present
Bit 1	Band 15 is present
Bit 29	Band 43 is present
Bit 30	Reserved
Bit 31	Reserved

Table 10.14 – Encoding of the Extra BandPresent parameter

Bitnumber	Meaning
Bit 0	ExtraBand 1 is present
Bit 1	ExtraBand 2 is present
…	…
Bit 29	ExtraBand 30 is present
Bit 30	ExtraBand 31 is present
Bit 31	ExtraBand 32 is present

Table 10.15 – Values for the setting of the Gain parameter

Value	corresponds to
$7F_{16}$	invalid
$7E_{16}$	+31.50 dB
…	…
00_{16}	0.00 dB
…	…
82_{16}	-31.50 dB
81_{16}	-31.75 dB
80_{16}	-32.00 dB

10.3.9 Automatic Gain Control

The Automatic Gain Control (AGC) is one of the optional building blocks of a *feature function block*. An Automatic Gain Control can have only the current setting attribute (CURRENT). The position of an Automatic Gain Control CURRENT attribute can be either TRUE (70_{16}) or FALSE (60_{16}). Also the value FF_{16} indicates invalidity and shall not be used for a control command..

In the first form of the *FUNCTION BLOCK command*, a particular Automatic Gain Control within a *feature function block* is addressed through the function_block_ID and *audio_channel_number* fields of the FUNCTION BLOCK command. The valid range for the *audio_channel_number* field is from zero (the 'master' channel) up to the number of logical channels in the audio channel cluster.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (1)							
Operand[7]	AGC_On							

Figure 10.32 – First Form of the AGC Control Parameters

Operand[5]	AGC_CONTROL
Operand[6]	1

Figure 10.33 – Values for the fields in the first form of the Mute Control Parameters

In the second form, the *audio_channel_number* field is set to FF_{16}. The parameter block contains a list of settings for all available AGC Controls in the feature function block.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (n)							
Operand[6+1]	AGC_On[1]							
Operand[6+n]	AGC_On[n]							

Figure 10.34 – Second Form of the AGC Control Parameters

Operand[5]	AGC_CONTROL
Operand[6]	Number of available Controls: NrAv

Figure 10.35 – Field values in the second form of the AGC Control Parameter Block

10.3.10 Delay Control

The Delay Control is one of the optional building blocks of a *feature function block*. A Delay Control can support following Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The settings for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes can range from zero (0000_{16}) to 1023.9687ms ($FFFE_{16}$) in steps of 1/64 ms or 0.015625 ms (0001_{16}). The RESOLUTION attribute can only take positive values and range from 01_{16} 1/64 ms or .015625 ms up to $7E_{16}$ which would be a resolution of 1.98375 ms. The Delay Control honors the command to the best of its abilities. It may round the Delay_On attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

In the first form of the FUNCTION BLOCK command, a particular Delay Control within a feature function block is addressed through the function_block_ID and *audio_channel_number* fields of the FUNCTION BLOCK command. The valid range for the *audio_channel_number* field is from zero (the 'master' channel) up to the number of logical channels in the audio channel cluster.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (2)							
Operand[6+1]	Delay_On_High							
Operand[6+2]	Delay_On_Low							

Figure 10.36 – First Form of the Delay Control Parameter Block

Operand[5]	DELAY_CONTROL
Operand[6]	2

Figure 10.37 – Values for the fields in the first form of the Delay Control Parameters

Table 10.16 – Values for the setting of the Delay attribute

Value	corresponds to
0000_{16}	0.0000 ms
0001_{16}	0.0156ms
0002_{16}	0.0312ms
…	…
0040_{16}	1.0000ms
…	…
$FFFD_{16}$	1023.9531ms
$FFFE_{16}$	1023.9687ms
$FFFF_{16}$	invalid

In the second form, the *audio_channel_number* field is set to FF_{16}. The parameter block contains a list of settings for all available Delay Controls in the Feature function block.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (2*n)							
Operand[6+1]	Delay_On_High[1]							
Operand[6+2]	Delay_On_Low[1]							
…	…							
…	…							
Operand[6+2*n-1]]	Delay_On_High[n]							
Operand[6+2*n]	Delay_On_Low[n]							

Figure 10.38 – Second Form of the Delay Control Parameter Block

Operand[5]	DELAY_CONTROL
Operand[6]	(Number of available Controls: NrAv)*2

Figure 10.39 – Values for the fields in the first form of the Delay Control Parameters

10.3.11 Bass Boost Control

The Bass Boost Control is one of the optional building blocks of a *feature function block*. A Bass Boost Control can have only the current setting attribute (CURRENT). The position of a Bass Boost Control CURRENT attribute can be either TRUE (70_{16}) or FALSE (60_{16}). Also the value FF_{16} indicates invalidity and shall not be used for a control command.

In the first form of the FUNCTION BLOCK command, a particular Bass Boost Control within a *feature function block* is addressed through the function_block_ID and *audio_channel_number* fields of the *feature function block command*. The valid range for the *audio_channel_number* field is from zero (the 'master' channel) up to the number of logical channels in the audio channel cluster.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (1)							
Operand[7]	BassBoost_On							

Figure 10.40 – First Form of the Bass Boost Control Parameters

Operand[5]	BASS_BOOST_CONTROL
Operand[6]	1

Figure 10.41 – Values for the fields in the first form of the Bass Boost Control Parameters

In the second form, the *audio_channel_number* field is set to FF_{16}. The parameter block contains a list of settings for the CURRENT attribute for all available Bass Boost Controls in the Feature function block.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (n)							
Operand[6+1]	BassBoost_On[1]							
Operand[6+n]	BassBoost_On[n]							

Figure 10.42 – Second Form of the Bass Boost Control Parameters

Operand[5]	BASS_BOOST_CONTROL
Operand[6]	Number of available Controls: NrAv

Figure 10.43 – Values for the second form of the Bass Boost Control Parameter Block

10.3.12 Loudness Control

The Loudness Control is one of the building blocks of a feature function block. A Loudness Control can have only the current setting attribute (CURRENT). The position of a Loudness Control CURRENT attribute can be either TRUE (70_{16}) or FALSE (60_{16}). Also the value FF_{16} indicates invalidity and shall not be used for a control command.

In the first form of the *function-block (audio) control command*, a particular Loudness Control within a *feature function block* is addressed through the function_block_ID and *audio_channel_number* fields of the *FUNCTION BLOCK command for feature function block*. The valid range for the *audio_channel_number* field is from zero (the 'master' channel) up to the number of logical channels in the audio channel cluster.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (1)							
Operand[7]	Loudness_On							

Figure 10.44 – First Form of the Loudness Control Parameters

Operand[5]	LOUDNESS_CONTROL
Operand[6]	1

Figure 10.45 – Values for the fields in the first form of the Loudness Control Parameters

In the second form, the *audio_channel_number* field is set to FF_{16}. The parameter block contains a list of settings for the CURRENT attribute for all available Loudness Controls in the Feature function block.

	msb							lsb
Operand[5]	Control Selector							
Operand[6]	length of control data (n)							
Operand[6+1]	Loudness_On[1]							
Operand[6+n]	Loudness_On[n]							

Figure 10.46 – Second Form of the Loudness Control Parameters

Operand[5]	LOUDNESS_CONTROL
Operand[6]	Number of available Controls: NrAv

Figure 10.47 – Values for the fields in the second form of the Loudness Control Parameter Block

10.4 Processing function block

The following paragraphs describe the *FUNCTION BLOCK commands for processing function block*. They are used to manipulate the audio controls within a Processing function block. This command is used to modify/retrieve an audio control inside a particular *processing function block* of the audio subunit.

	msb							lsb
Operand[3]	4							
Operand[4]	fb-input plug number (FBPN)							
Operand[5]	input audio channel number (ICN)							
Operand[6]	output audio channel number (OCN)							
Operand[7]	control_selector							
Operand[7+1] ... Operand[7+n]	processing_function_block_type_dependent_parameters							

Figure 10.48 – Processing FUNCTION BLOCK command

The *selector_length* field (Operand[3]) for processing function block shall always be set to 4. The *audio_selector_data* shall consist of *fb-input_plug_number* (Operand4] *input_audio_channel_number (ICN)* (Operand[5]), *output_audio_channel_number (OCN)* (Operand[6]) and *control_selector* (Operand[7]).

The processing function block can have one or more input fb-plugs. If it has only one input fb-plug, the operand[4] field shall be specified 00_{16}

The input and output *audio_channel_numbers* determine the individual controls being addressed. Processing function blocks may have controls which affect the audio within a single channel: OCN = output *audio_channel_number* field and ICN = FE_{16}. Except for the FUNCTION BLOCK command

issued to the processing function block (mixer), the input *audio_channel_number* (ICN) field is specified FE_{16}. The value FE_{16} (254) indicates the input *audio_channel_number* is "VOID" for this control_selector in the processing function block. Only processing function block (mixer) has a valid input *audio_channel_number*.

If the input *audio_channel_number* field is specified FE_{16} and the output *audio_channel_number* field is specified 00_{16} this indicates the "master" controls which affect the master channel for the output audio channels. Output *audio_channel*_number may range from 1 to the highest number of the valid channel which is implemented. And it may be 00_{16} (indicating the "master channel"), FE_{16}, or FF_{16}. When specifying the output *audio_channel_number* is no meaning (e.g. *control_selector*=MODE_CONTROL), the output *audio_channel_number* is FE_{16}. Mixer Type Processing Function Blocks may have controls which affect audio across channels (ICN ≠ OCN or ICN = OCN).

If the command specifies an unknown ICN, FBPN or OCN to that function block, the response frame must indicate return NOT IMPLEMENTED.

10.4.1 FUNCTION BLOCK Command for Processing Function Block

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (n)							
Operand[8+1] … Operand[8+n]	processing_function_block_type_dependent_parameters							

Figure 10.49 – Processing FUNCTION BLOCK command

The control_attribute field contains a constant, identifying which attribute of the addressed Control is to be modified. A Processing function block Control may support all possible control attributes:

- CURRENT
- MINIMUM
- MAXIMUM
- RESOLUTION
- DEFAULT

If the addressed Control does not support modification of a certain attribute, the response frame shall indicate NOT IMPLEMENTED when an attempt is made to modify that attribute. In most cases, only the CURRENT attribute will be supported for the FUNCTION BLOCK command for processing function block. However, this specification does not prevent a designer from making other attributes programmable.

The operand[7] as shown above in Figure 10.49 specifies the *control_selector*. The *control_selector* indicates which type of control this command is manipulating. (Enable Processing, Mode Select, etc.) If the command specifies an unknown or unsupported *control_selector* to that Processing Function block, the response frame shall indicate NOT IMPLEMENTED.

The operand [1] as shown in Figure 10.1, specifies the *function_block_ID* (FBID). Only existing *processing function blocks* in the audio function can be addressed. If the command specifies an unknown or non-existing FBID, the response frame shall indicate NOT IMPLEMENTED.

The actual parameter(s) for the processing function block control are passed in the processing function block dependent parameters. The length of the parameter block is indicated in the operand[8] field of the FUNCTION BLOCK command.

10.4.2 Processing Function Block Controls

Processing function block controls are in principal specific to the type of process a *processing function block* implements. In addition, in case support for a particular Control is optional for a certain processing function block, the Controls field of the processing function block descriptor indicates what controls the *processing function block* supports. In this way, controls that occur in different *processing function blocks* need only be specified once. Issuing non-supported Control Selectors to a *processing function block* leads to the response frame indicating NOT IMPLEMENTED.

The following paragraphs present a detailed description of all possible controls a *processing function block* can incorporate. For each Control, the layout of the parameter block together with the appropriate Control Selector is listed for all forms of the processing *FUNCTION BLOCK command*.

10.4.2.1 Enable Processing Control

The Enable Processing Control is used to either enable the functionality of the *processing function block* or bypass the *processing function block* entirely. In the latter case, default behavior is assumed. The position of the Enable Processing switch can be either TRUE (70_{16}) or FALSE (60_{16}). Also the value FF_{16} indicates invalidity and shall not be used for a control command.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control data (1)							
Operand[8+1]	Processing_On							

Figure 10.50 – Enable Processing Control Parameter Block

Operand[7]	ENABLE_PROCESSING_CONTROL
Operand[8]	1

Figure 10.51 – Values for the fields in the first form of the Enable Processing Control Parameters

10.4.2.2 Mode Select Control

The Mode Select Control is used to change the behavior of the Processing Function block. The Mode Select Control range is from one to the number of modes, supported by the Processing Function block (reported through the number_of_modes field of the Processing Function block descriptor). The current mode can be queried using FUNCTION BLOCK command for processing function block.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control data (m+1)							
Operand[9]	Size_of_modes(m)							
Operand[9+1]	Mode[1]							
…	…							
Operand[9+m]	Mode[n]							

Figure 10.52 – Mode Control Parameter Block

Operand[7]	MODE_CONTROL
Operand[8]	Size_of_Modes+1
Operand[9]	Size_of_Modes

Figure 10.53 – Mode Select Control Parameter Block

The setting for the Mode Select Control can range from 1 to *number_of_modes*.

The FUNCTION BLOCK command with a *control_selector* of MODE_CONTROL may be used as status command. This command has a *ctype* value of STATUS. In this case, each field of *Mode operand* is set to FF_{16} when the status command is issued and is updated to the current setting of the control_attribute when the STABLE response frame is returned.

None of the fields of *Mode operand* in the control command shall be set to FF_{16}.

10.4.3 Function Block Command for mixer processing function block

The attributes of a ***mixer*** *control* inside a *mixer processing function block* of the audio subunit can be set by filling in the **ctype**-field of the audio command with the value “CONTROL”.

	msb							lsb
Operand[3]	4							
Operand[4]	fb-inputplug number (FBPN)							
Operand[5]	input audio channel number (ICN)							
Operand[6]	output audio channel number (OCN)							
Operand[7]	Control_selector							
Operand[7+1] … Operand[7+n]	Control_data							

Figure 10.54 – FUNCTION BLOCK command for mixer processing function block

The **control_attribute** field of the command (operand[2]) contains a constant, identifying which attribute of the control is to be addressed. Possible attributes for a *mixer processing function block control* are:

— CURRENT

— MINIMUM

— MAXIMUM

— RESOLUTION

— DEFAULT

If the addressed control does not support modification or readout of a certain attribute, the response frame must return NOT IMPLEMENTED when an attempt is made to modify that attribute. In most cases, only the CURRENT attribute will be supported for the *FUNCTION BLOCK command for mixer processing function block*. However, this specification does not prevent a designer from making other attributes programmable.

The operand[5] field specifies the Input *audio_channel_number* (ICN) to be addressed. The input fb-plug containing this input channel is indicated in the operand[4] field. The Output *audio_channel_number* (OCN) of the mixer control to be influenced is indicated in operand[6].

The operand[1] field specifies the *function_block_ID* (FBID). Only existing Mixer function blocks in the audio function can be addressed. If the command specifies an unknown or non-mixer FBID the response frame will return NOT IMPLEMENTED.

If the command specifies an unknown ICN, FBPN or OCN to that function block or refers to a non-programmable mixer control in the mixer function block, the response frame must return NOT IMPLEMENTED.

A special case arises when the input *audio_channel_number* and output *audio_channel_number* are both set to FF_{16}. Then a single FUNCTION BLOCK command can be used to modify/retrieve an attribute of all the programmable mixer controls within the function block. The number of parameters passed in the parameter block must exactly match the number of programmable mixer controls in the function block. If this is not the case, the response frame will return NOT IMPLEMENTED. The ordering of the parameters in the parameter block obeys the same rules as established for the bit ordering in the controls field of the mixer function block dependent data. The parameter block must contain an attribute setting for every Mixer Control that has its bit set in the controls field.

Another special case arises when the Input *audio_channel_number* and Output *audio_channel_number* are both set to 00_{16}. Then a single FUNCTION BLOCK command can be used to modify/retrieve an attribute setting of all the Mixer Controls (both programmable and non-programmable) within the function block. The ordering of the parameters in the parameter block obeys the same rules as established for the bit ordering in the Controls field of the mixer processing function block descriptor. The parameter block now contains a setting for every Mixer in the Mixer function block. The above description is referred to as the third form of the FUNCTION BLOCK command.

The length of the parameter block returned in the response frame is indicated in the operand[8] field of the command. If the control setting(s) are longer than the operand[8] field, only the initial bytes of the control setting(s) are returned. If the control setting(s) are shorter than the operand[8] field, the response frame will only return enough data as is necessary.

10.4.4 Mixer Control

A mixer processing function block consists of a number of Mixer Controls, either programmable or fixed. A Mixer Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The settings for the CURRENT, MINIMUM, MAXIMUM and DEFAULT attributes can range from +127.9922dB ($7FFE_{16}$) down to–-127.9961 dB (8001_{16}) in steps of 1/256 dB or 0.00390625 dB (0001_{16}), Code 8000_{16} represents silence, i.e. -∞ dB. The settings for the RESOLUTION attribute can only take positive values and range from 1/256 dB (0001_{16}) to +127.9922dB ($7FFE_{16}$). The Mixer Control honors the command to the best of its abilities. It may round the Mixer field value to its closest available setting. It will report this rounded setting when queried during a FUNCTION BLOCK command.

In the first form of the FUNCTION BLOCK command for mixer processing function block, a particular Mixer Control within a function block is addressed through the function block ID, Input *audio_channel_number* and Output *audio_channel_number* fields of the FUNCTION BLOCK command.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	length of control data (2)							
Operand[9]	Mixer_Setting_High							
Operand[10]	Mixer_Setting_Low							

Figure 10.55 – First from of the mixer control parameters

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	length of control data (2)							
Operand[9]	Mixer_Setting_High							
Operand[10]	Mixer_Setting_Low							

Figure 10.56 – First from of the mixer control parameters status form of the command

Table 10.17 – Table of mixer settings

Mixer setting	corresponds to
$7FFF_{16}$	invalid
$7FFE_{16}$	127.9922dB
…	…
0100_{16}	1.0000dB
…	…
0002_{16}	0.0078dB
0001_{16}	0.0039dB
0000_{16}	0.0000dB
$FFFF_{16}$	-0.0039dB
$FFFE_{16}$	-0.0078dB
…	…
$FF00_{16}$	-1.0000dB
…	…
8002_{16}	-127.9922dB
8001_{16}	-127.9961dB
8000_{16}	-∞dB

Operand[7]	MIXER_CONTROL
Operand[8]	2

Figure 10.57 – Values for the fields in the first form of the Mixer Control Parameter Block

In the second form, the Input and Output *audio_channel_number* fields are both set to FF_{16}. The parameter block contains a list of settings for an attribute of all programmable Mixer Controls in the mixer processing function block. The length of control data is given in bytes.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	length of control data (n*2)							
Operand[8+1]	Programmable_Mixer_Setting_High(1)							
Operand[8+2]	Programmable_Mixer_Setting_Low(1)							
…	…							
…	…							
Operand[8+2*n-1]	Programmable_Mixer_Setting_High(n)							
Operand[8+2*n]	Programmable_Mixer_Setting_Low(n)							

Figure 10.58 – Second Form of the Mixer Control Parameter Block

Operand[7]	MIXER_CONTROL
Operand[8]	(Number of programmable Controls: NrPr)*2

Figure 10.59 – Field values in the second form of the Mixer Control Parameter Block

In the third form, the Input and Output *audio_channel_number* fields are both set to 00_{16}. The parameter block contains a list of settings for an attribute of all the Mixer Controls in the mixer processing function block. The length of control data is given in Bytes.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	length of control data (n*2)							
Operand[8+1]	Mixer_Setting_High(1)							
Operand[8+2]	Mixer_Setting_Low(1)							
…	…							
…	…							
Operand[8+2*n-1]	Mixer_Setting_High(n)							
Operand[8+2*n]	Mixer_Setting_Low(n)							

Figure 10.60 – Third Form of the Mixer Control Parameter Block

Operand[7]	MIXER_CONTROL
Operand[8]	(Number of Controls: NrCo)*2

Figure 10.61 – Field values for the third form of the Mixer Control Parameter Block

10.5 Controls Specific to particular processing function types

10.5.1 Spaciousness Control

The Spaciousness Control is used to change the spatial appearance of the stereo image, produced by the *3D-Stereo extender processing function block*. The Spaciousness Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The valid range for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes is from zero to 254%. Also the value FF_{16} indicates invalidity and shall not be used for a control command. The Spaciousness Control honors the request to the best of its abilities. It may round the **Spaciousness** attribute value to its closest available setting. It will report this rounded setting when queried during by a FUNCTION BLOCK command.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control data (1)							
Operand[8+1]	Spaciousness							

Figure 10.62 – Spaciousness Control Parameter Block

Operand[7]	SPACIOUSNESS_CONTROL
Operand[8]	1

Figure 10.63 – Values for Spaciousness Parameters

10.5.2 Reverberation Type Control

The Reverberation Type Control is a macro parameter that allows global settings of reverberation parameters within the *reverberation processing function block*. When a certain Reverberation Type is selected, each reverberation parameter will be set to the most suitable value. The Reverberation Type Control can support the CURRENT and, DEFAULT Control. The valid range for the CURRENT, and DEFAULT attributes is from zero to 254. Also the value FF_{16} indicates invalidity and shall not be used for a control command.

The CURRENT attribute subrange from 0 to 7 has predefined behavior:

0: Room 1 – simulates the reverberation of a small room.

1: Room 2 – simulates the reverberation of a medium room.

2: Room 3 – simulates the reverberation of a large room.

3: Hall 1 – simulates the reverberation of a medium concert hall.

4: Hall 2 – simulates the reverberation of a large concert hall.

5: Plate – simulates a plate reverberation (a studio device using a metal plate).

6: Delay – conventional delay that produces echo effects.

7: Panning Delay – special delay in which the delayed sounds move left and right.

The Reverberation Type Control honors the request to the best of its abilities. It may round the **ReverbType** attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (1)							
Operand[8+1]	ReverbType							

Figure 10.64 – Reverberation Type Control Parameter Block

Operand[7]	REVERBTYPE_CONTROL
Operand[8]	1

Figure 10.65 – Values for ReverbType Parameters

The setting for the attribute of the Reverberation Type Control is chosen between the possibilities outlined above.

10.5.3 Reverberation Level Control

The Reverberation Level Control is used to set the amount of reverberant sound introduced by the Reverberation Processing function block. The Reverberation Level Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The valid range for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes is from zero to 254%, compared to the level of the original signal. Also the value FF_{16} indicates invalidity and shall not be used for a control command. The Reverberation Level Control honors the request to the best of its abilities. It may round the **ReverbLevel** attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (1)							
Operand[8+1]	ReverbLevel							

Figure 10.66 – First form of the Reverberation Level Control Parameter Block

Operand[7]	REVERBLEVEL_CONTROL
Operand[8]	1

Figure 10.67 – Values for the first form of the Reverberation Level Parameters

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (n)							
Operand[8+1]	ReverbLevel[1]							
…	…							
Operand[8+n]	ReverbLevel[n]							

Figure 10.68 – Second form of the Reverberation Level Control Parameter Block

Operand[7]	REVERBLEVEL_CONTROL
Operand[8]	nchannels

Figure 10.69 – Values for the second form of the Reverberation Level Parameters

10.5.4 Reverberation Time Control

The Reverberation Time Control is used to set the time over which the reverberation, introduced by the Reverberation Processing function block, will continue. The Reverberation Time Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The settings for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes can range from zero (0000_{16}) to 255.9922 s ($FFFE_{16}$) in steps of 1/256 s or 0.00390625 s (0001_{16}). Also the value $FFFF_{16}$ indicates invalidity and shall not be used for a control command.The Reverberation Time Control honors the request to the best of its abilities. It may round the **ReverbTime** attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (2)							
Operand[7+1]	ReverbTime_High							
Operand[8+2]	ReverbTime_Low							

Figure 10.70 – First form of Reverberation time Control Parameter block

Operand[7]	REVERBTIME_CONTROL
Operand[8]	2

Figure 10.71 – Values for first form of the ReverbTime Parameters

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (2*n)							
Operand[8+1]	ReverbTime_High[1]							
Operand[8+2]	ReverbTime_Low[1]							
	…							
Operand[8+2*n-1]	ReverbTime_High[n]							
Operand[8+2*n]	ReverbTime_Low[n]							

Figure 10.72 – Second form of the Reverberation Time Control Parameter Block

Operand[7]	REVERBTIME_CONTROL
Operand[8]	2*nchannels

Figure 10.73 – Values for the second form of the Reverberation Time Parameters

10.5.5 Reverberation Early Time Control

The Reverberation Early Time Control is used to set the time after which the first reflection, introduced by the Reverberation Processing function block, will arrive. The Reverberation Early Time Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The settings for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes can range from zero (0000_{16}) to 255.9922 s ($FFFE_{16}$) in steps of 1/256 s or 0.00390625 s (0001_{16}). Also the value $FFFF_{16}$ indicates invalidity and shall not be used for a control command. The Reverberation Early Time Control honors the request to the best of its abilities. It may round the **ReverbEarly** attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

	Msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (2)							
Operand[8+1]	ReverbEarlyTime_High							
Operand[8+2]	ReverbEarlyTime_Low							

Figure 10.74 – First form of Reverberation Early Time Control Parameter block

Operand[7]	REVERB_EARLYTIME_CONTROL
Operand[8]	2

Figure 10.75 – Values for first form of the Reverberation Early Time Parameters

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (2*n)							
Operand[8+1]	ReverbEarlyTime_High[1]							
Operand[8+2]	ReverbEarlyTime_Low[1]							
	…							
Operand[8+2*n-1]	ReverbEarlyTime_High[n]							
Operand[8+2*n]	ReverbEarlyTime_Low[n]							

Figure 10.76 – Second form of the Reverberation Early Time Control Parameter Block

Operand[7]	REVERB_EARLYTIME_CONTROL
Operand[8]	2*nchannels

Figure 10.77 – Values for the second form of the Reverberation Early Time Parameters

10.5.6 Reverberation Delay Feedback Control

The Reverberation Delay Feedback Control is used when the Reverberation Type is set to Reverberation Type 6, Delay or Reverberation Type 7, Panning Delay. It sets the way in which delay repeats. The Reverberation Delay Feedback Control range can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The valid range for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes is from zero to 254%. Also the value FF_{16} indicates invalidity and shall not be used for a control command. Higher values result in more delay repeats[2].

The Reverberation Delay Feedback Control honors the request to the best of its abilities. It may round the **ReverbFeedback** attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (1)							
Operand[8+1]	ReverbFeedback							

Figure 10.78 – Reverberation Delay Feedback Control Parameter Block

Operand[7]	REVERBFEEDBACK_CONTROL
Operand[8]	1

Figure 10.79 – Values for Reverberation Delay Parameters

10.5.7 Chorus Level Control

The Chorus Level Control is used to set the amount of chorus effect sound introduced by the Chorus Processing function block. The Chorus Level Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The valid range for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes is from zero to 254%, compared to the level of the original signal. Also the value FF_{16} indicates invalidity and shall not be used for a control command. The Chorus Level Control honors the request to the best of its abilities. It may round the **ChorusLevel** attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

	Msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (1)							
Operand[8+1]	ChorusLevel							

Figure 10.80 – Chorus Level Control Parameter Block

[2] In practice, the delay feedback amount should be limited to 75% to avoid unexpected feedback distortion and continuous delay loop.

Operand[7]	CHORUSLEVEL_CONTROL
Operand[8]	1

Figure 10.81 – Values for Chorus Level Parameters

10.5.8 Chorus Modulation Rate Control

The Chorus Modulation Rate Control is used to set the speed (frequency) of the modulator of the chorus, introduced by the Chorus Processing function block. Higher values result in faster modulation. The Chorus Modulation Rate Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The settings for the CURRENT, MINIMUM, MAXIMUM, DEFAULT, RESOLUTION and DELTA attributes can range from zero (0000_{16}) to 255.9922 Hz ($FFFE_{16}$) in steps of 1/256 Hz or 0.00390625 Hz (0001_{16}). Also the value $FFFF_{16}$ indicates invalidity and shall not be used for a control command. The Chorus Modulation Rate Control honors the request to the best of its abilities. It may round the **ChorusRate** attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (2)							
Operand[8+1]	ChorusRate_High							
Operand[8+2]	ChorusRate_Low							

Figure 10.82 – Chorus Modulation Rate Control Parameter Block

Operand[7]	CHORUSRATE_CONTROL
Operand[8]	2

Figure 10.83 – Values for Chorus Modulation Rate Parameters

10.5.9 Chorus Modulation Depth Control

The Chorus Modulation Depth Control is used to set the depth at which the chorus sound introduced by the Chorus Processing function block, is modulated. Higher values result in deeper modulation. The Chorus Modulation Depth Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The settings for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes can range from zero (0000_{16}) to 255.9922 ms ($FFFE_{16}$) in steps of 1/256 ms or 0.00390625 ms (0001_{16}). Also the value $FFFF_{16}$ indicates invalidity and shall not be used for a control command. The Chorus Modulation Depth Control honors the request to the best of its abilities. It may round the *ChorusDepth* attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (2)							
Operand[8+1]	ChorusDepth_High							
Operand[8+2]	ChorusDepth_Low							

Figure 10.84 – Chorus Modulation Depth Control Parameter Block

Operand[7]	CHORUSDEPTH_CONTROL
Operand[8]	2

Figure 10.85 – Values for Chorus Modulation Depth Parameters

10.5.10 Dynamic Range Compressor Compression Ratio Control

The Compression Ratio Control is used to influence the slope of the active part of the static input-to-output transfer characteristic of the Dynamic Range Compressor Processing Function Block. The Compression Ratio Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The valid range for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes is from zero (0000_{16}) to 255.9922 dB/dB ($FFFE_{16}$) in steps of 1/256 dB/dB or 0.00390625 dB/dB (0001_{16}). Also the value $FFFF_{16}$ indicates invalidity and shall not be used for a control command. The Compression Ratio Control honors the request to the best of its abilities. It may round the **Ratio** attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

	Msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (2)							
Operand[8+1]	CompressionRatio_High							
Operand[8+2]	CompressionRatio_Low							

Figure 10.86 – Dynamic Range Compressor Ratio Control Parameter Block

Operand[7]	COMPRESSION_RATIO_CONTROL
Operand[8]	2

Figure 10.87 – Values for Range Compression Ratio Parameters

10.5.11 Dynamic Range Compressor MaxAmpl Control

The MaxAmpl Control is used to set the upper boundary of the active input range of the compressor. The MaxAmpl Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The settings for the CURRENT, MINIMUM, DEFAULT, and MAXIMUM attributes can range from –128.0000 dB (8000_{16}) to 127.9922 dB ($7FFE_{16}$) in steps of 1/256 dB or 0.00390625 dB (0001_{16}). The settings for the RESOLUTION attribute can only take positive values and range from 1/256 dB (0001_{16}) to +127.9922 dB ($7FFE_{16}$). Also the value $7FFF_{16}$ indicates invalidity and shall not be used for a control command. The MaxAmpl Control honors the request to the best of its abilities. It may round the **MaxAmpl** attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (2)							
Operand[8+1]	MaxAmpl_High							
Operand[8+2]	MaxAmpl_Low							

Figure 10.88 – Dynamic Range Compressor MaxAmpl Control Parameter Block

Operand[7]	MAXAMPL_CONTROL
Operand[8]	2

Figure 10.89 – Values for MaxAmpl Parameters

10.5.12 Dynamic Range Compressor Threshold Control

The Threshold Control is used to set the lower boundary of the active input range of the compressor. The Threshold Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The settings for the CURRENT, MINIMUM, DEFAULT, and MAXIMUM attributes can range from –128.0000 dB (8000_{16}) to 127.9922 dB ($7FFE_{16}$) in steps of 1/256 dB or 0.00390625 dB (0001_{16}). The settings for the RESOLUTION attribute can only take positive values and range from 1/256 dB (0001_{16}) to +127.9922 dB ($7FFE_{16}$). Also the value $7FFF_{16}$ indicates invalidity and shall not be used for a control command. The Threshold Control honors the request to the best of its abilities. It may round the **Threshold** attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (2)							
Operand[8+1]	Threshold_High							
Operand[8+2]	Threshold_Low							

Figure 10.90 – Dynamic Range Compressor Threshold Control Parameter Block

Operand[7]	THRESHOLD_CONTROL
Operand[8]	2

Figure 10.91 – Values for Threshold Parameters

10.5.13 Dynamic Range Compressor Attack Time Control

The Attack Time Control is used to determine the response of the compressor to a step increase in the input signal level. The Attack Time Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The settings for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes can range from zero (0000_{16}) to 255.9922 ms ($FFFE_{16}$) in steps of 1/256 ms or 0.00390625 ms (0001_{16}). Also the value $FFFF_{16}$ indicates invalidity and shall not be used for a control command. The Attack Time Control honors the request to the best of its abilities. It may round the **AttackTime** attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (2)							
Operand[8+1]	AttackTime_High							
Operand[8+2]	AttackTime_Low							

Figure 10.92 – Dynamic Range Compressor Attack Time Control Parameter Block

Operand[7]	ATTACKTIME_CONTROL
Operand[8]	2

Figure 10.93 – Values for AttackTime Parameters

10.5.14 Dynamic Range Compressor Release Time Control

The Release Time Control is used to determine the recovery response of the compressor to a step decrease in the input signal level. The Release Time Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The settings for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes can range from zero (0000_{16}) to 255.9922 ms ($FFFE_{16}$) in steps of 1/256 ms or 0.00390625 ms (0001_{16}). Also the value $FFFF_{16}$ indicates invalidity and shall not be used for a control command. The Release Time Control honors the request to the best of its abilities. It may round the **ReleaseTime** attribute value to its closest available setting. It will report this rounded setting when queried by a FUNCTION BLOCK command.

	msb							lsb
Operand[7]	Control Selector							
Operand[8]	Length of control parameters (2)							
Operand[8+1]	ReleaseTime_High							
Operand[8+2]	ReleaseTime_Low							

Figure 10.94 – Dynamic Range Compressor Release Time Control Parameter Block

Operand[7]	RELEASETIME_CONTROL
Operand[8]	2

Figure 10.95 – Values for MaxAmpl Parameters

10.6 FUNCTION BLOCK commands for CODEC function block

The following paragraphs detail the layout of the FUNCTION BLOCK command when it is addressed to a CODEC function block. The command will contain data that are specific to a certain audio format and depend on the control that is addressed within the function block.

	msb							lsb
Operand[3]	1							
Operand[4]	Control Selector							
Operand[5]	Length of control data (n)							
Operand[5+1] … Operand[5+n]	control_data							

Figure 10.96 – CODEC audio command further detailed

Operand[3], *selector_length* shall be set to 1 because there is no *audio_selector_data.*

10.6.1 CODEC Function Block Controls

CODEC function block controls are in principal specific to the type of decoding a *CODEC function block* implements. In addition, in case support for a particular Control is optional for a certain processing function block, the Controls field of the processing function block descriptor indicates what controls the *CODEC function block* supports. In this way, controls that occur in different *CODEC function blocks* need only be specified once. Issuing non-supported Control Selectors to a *CODEC function block* leads to the response frame indicating NOT IMPLEMENTED.

The following paragraphs present a detailed description of all possible controls a *CODEC function block* can incorporate. For each Control, the layout of the parameter block together with the appropriate Control Selector is listed for all forms of the *FUNCTION BLOCK command for CODEC function block*.

A CODEC should support the enable processing and mode select controls.

10.6.2 Enable CODEC Control

The Enable CODEC Control is used to either enable the functionality of the *CODEC function block* or bypass the *CODEC function block* entirely. In the latter case, default behavior is assumed. The position of the Enable Processing switch can be either TRUE or FALSE.

	msb							lsb
Operand[4]	Control Selector							
Operand[5]	Length of control parameters (1)							
Operand[5+1]	CODEC_On							

Figure 10.97 – Enable CODEC Control Parameter Block

Operand[4]	ENABLE_CONTROL
Operand[5]	1

Figure 10.98 – Values for the fields in the first form of the Enable CODEC Control Parameters

10.6.3 Mode Select Control

The Mode Select Control is used to change the behavior of the CODEC Function block. The Mode Select Control range is from one to the number of modes, supported by the Processing Function block (reported

through the NrModes field of the CODEC Function block descriptor). The current mode can be queried using a CODEC Function block Control command.

	msb							lsb
Operand[4]	Control Selector							
Operand[5]	Length of control parameters (n)							
Operand[5+1]	Mode[1]							
…	…							
Operand[5+n]	Mode[n]							

Figure 10.99 – Mode Control Parameter Block

Operand[4]	MODE_CONTROL
Operand[5]	1

Figure 10.100 – Mode Select Control Parameter Block

The setting for the Mode Select Control. Can range from 1 to NrModes.

10.6.4 Controls for Specific Types of CODEC

10.6.4.1 DTS Format-Specific Controls

There are no controls specific to the DTS CODEC.

10.6.4.2 MPEG Format-Specific Controls

The following paragraphs describe the MPEG audio control specific to MPEG formats. Some of the control parameters are also dependent on the content of the incoming MPEG data stream.

In general, the behavior of the MPEG decoder is primarily controlled by the incoming bit-stream. Parameters modified using a FUNCTION BLOCK command manipulating the MPEG attributes retain their setting, even if that setting is not applicable to the current incoming bit-stream.

As an example, consider a decoder that is receiving a stream containing two independent stereo channel pairs. In this case, the Select Second Stereo Control can be enabled so that the second stereo channel is reproduced over the Left and Right channel. If the incoming stream is now switched to a full 5.1 encoded stream, the Select Second Stereo Control has no more influence and the decoder overrides its setting and produces full 5.1 sound. However, if the incoming stream switches back to the previous format, the Select Second Stereo Control becomes active again and resumes its previous setting so that the second stereo channel is reproduced again over the Left and Right channel.

The following paragraphs present a detailed description of all possible MPEG Controls a CODEC function block can incorporate. For each Control, the layout of the parameter block together with the appropriate Control Selector is listed.

10.6.4.2.1 Dual Channel Control

The Dual Channel Control is used to select which of the two available stereo pairs. in the MPEG-1 base stream is actually retrieved and reproduced over the Left and Right output channels. If this Control is addressed on a decoder that does not implement Dual Channel Control (D4 = '0' in the **MPEGCapabilities** field of the MPEG format-specific descriptor), the response frame must indicate NOT IMPLEMENTED.

The Dual Channel Control can have only the current setting attribute (CURRENT). The position of the Channel2Enable switch can be either TRUE (70_{16}) or FALSE (60_{16}). Also the value FF_{16} indicates invalidity and shall be not used for a control command. When FALSE, Channel I is selected, and when TRUE, Channel II is selected. The current setting of the Control can be queried using a Get MPEG Control request.

	msb							lsb
Operand[4]	Control Selector							
Operand[5]	Length of control data (1)							
Operand[5+1]	Dual_Channel_Enable							

Figure 10.101 – MPEG Dual Channel Control Parameter Block

The setting for the attribute of the Dual Channel Control: Channel I selected when FALSE, Channel II selected when TRUE.

Operand[4]	MPEG_DUAL_CHANNEL_CONTROL
Operand[5]	1

Figure 10.102 – Values for the fields of the MPEG Dual Channel Control Parameters

10.6.4.2.2 Second Stereo Control

The Second Stereo Control is used to select the second stereo channel pair that can be encoded in an MPEG-2 stream instead of the multi-channel stereophonic information (3/2). If this Control is addressed on a decoder that does not implement Second Stereo support (D5 = '0' in the **MPEGCapabilities** field of the MPEG format-specific descriptor), the response frame must indicate NOT IMPLEMENTED.

The Second Stereo Control can have only the current setting attribute (CURRENT). The position of the 2ndStereoEnable switch can be either TRUE (70_{16}) or FALSE (60_{16}). Also the value FF_{16} indicates invalidity and shall not be used for a control command. When FALSE, the main stereo channel pair is selected; when TRUE, the second stereo channel pair is selected. The current setting of the Control can be queried using a Get MPEG Control request.

	Msb							lsb
Operand[4]	Control Selector							
Operand[5]	Length of control data (1)							
Operand[5+1]	Second_Stereo_Enable							

Figure 10.103 – Second Stereo Control Parameter Block

Operand[4]	MPEG_SECOND_STEREO_CONTROL
Operand[5]	1

Figure 10.104 – Values for the fields in the 2nd stereo Control

10.6.4.2.3 Multilingual Control

The Multilingual Control is used to select the multilingual channel actually retrieved from the MPEG stream. If this Control is addressed on a decoder that does not implement multilingual support (D9..8 = '00' in the **MPEGCapabilities** field of the MPEG format dependent descriptor), the response frame must indicate NOT IMPLEMENTED.

The Multilingual Control supports only the CURRENT Control attribute. The valid range is from zero (00_{16}) to seven (07_{16}). The actual range depends on the incoming MPEG stream. It may contain only a limited number of multilingual channels (less than seven). The Multilingual Control honors the request to the best of its abilities. It may truncate the attribute values to its closest available settings. It will report these settings when queried with a FUNCTION BLOCK command (ctype = STATUS). Also the value FF_{16} indicates invalidity and shall not be used for a control command.

	msb							lsb
Operand[4]	Control Selector							
Operand[5]	Length of control parameters (1)							
Operand[5+1]	Multi_Lingual							

Figure 10.105 – Multilingual Control Parameter Block

The setting for the attribute of the multilingual channel selection:

- 0 = decode no channel
- 1..7 = decode channel 1..7
- 8..255 = reserved

Operand[4]	MPEG_MULTILINGUAL_CONTROL
Operand[5]	1

Figure 10.106 – Values for the fields of the Multilingual Control Parameters

10.6.4.2.4 Dynamic Range Control

The Dynamic Range Control (DRC) is used to enable or disable the Dynamic Range Control functionality of the decoder. If the decoder does not support Dynamic Range control (D5..4 = '00' in the **MPEGFeatures** field of the MPEG format-specific descriptor), the response frame must return NOT IMPLEMENTED.

The Dynamic Range Control can have only the current setting attribute (CURRENT). The position of the DRC switch can be either TRUE (70_{16}) or FALSE (60_{16}). Also the value FF_{16} indicates invalidity and shall not be used for a control command. TRUE means that the MPEG decoder is using the Dynamic Range control words (possibly with additional scaling) contained in the MPEG bit stream to control the audio dynamic range. FALSE means the control words are being ignored, and the original signal dynamic range is being reproduced. The current setting of the Control can be queried using a Get MPEG Control request.

	Msb							lsb
Operand[4]	Control Selector							
Operand[5]	Length of control parameters (1)							
Operand[5+1]	Dynamic_Range							

Figure 10.107 – Dynamic Range Control Parameter Block

Operand[4]	MPEG_DYN_RANGE_CONTROL
Operand[5]	1

Figure 10.108 – Values for the fields of the Dynamic Range Control Parameters

10.6.4.2.5 Scaling Control

The Scaling Control is used to manipulate the single scaling coefficient used by MPEG decoders that implement a common boost/cut scaling value for Dynamic Range Control (D5..4 = '10' in the **MPEGFeatures** field of the MPEG format-specific descriptor). If this Control is addressed on a non-'10' decoder, the response frame must return NOT IMPLEMENTED.

The Scaling Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The valid range for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes is from zero (00_{16}) to 254/256 (FE_{16}). Also the value FF_{16} indicates invalidity and shall not be used for a control command. The Scaling Control honors the request to the best of its abilities. It may round the **Scale** attribute value to its closest available setting. It will report this rounded setting when queried in an MPEG FUNCTION BLOCK command.

	Msb							lsb
Operand[4]	Control Selector							
Operand[5]	Length of control parameters (1)							
Operand[5+1]	Scale_Setting							

Figure 10.109 – Scaling Control Parameter Block

Operand[4]	MPEG_SCALE_CONTROL
Operand[5]	1

Figure 10.110 – Values for the fields of the Scaling Control Parameters

10.6.4.2.6 High/Low Scaling Control

The High/Low Scaling Control is used to manipulate the two scaling coefficients used by MPEG decoders that implement an independent boost and cut scaling value for Dynamic Range Control (D5..4 = '11' in the **MPEGFeatures** field of the MPEG format-specific descriptor). If this Control is addressed on a non-'11' decoder, the response frame must return NOT IMPLEMENTED.

The High/Low Scaling Control can support following Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The valid range for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes is from zero (00_{16}) to 254/256 (FE_{16}).

Also the value FF_{16} indicates invalidity and shall not be used for a control command.. The High/Low Scaling Control honors the request to the best of its abilities. It may round the **LowScale** and **HighScale** attribute values to their closest available settings. It will report these rounded settings when queried by an audio command. The **LowScale** value is used by the MPEG decoder to scale the Dynamic Range control words that apply a gain increase (for low sound levels). The **HighScale** value is used by the MPEG decoder to scale the Dynamic Range control words that apply a gain reduction (for high level sounds).

	msb							lsb
Operand[4]	Control Selector							
Operand[5]	Length of control parameters (2)							
Operand[5+1]	LowScale							
Operand[5+2]	HighScale							

Figure 10.111 – High/Low Scaling Control Parameter Block

Operand[4]	MPEG_HL_SCALING_CONTROL
Operand[5]	2

Figure 10.112 – Values for the fields for the High/Low Scaling Control Parameters

10.6.4.3 AC-3 Controls

The following paragraphs present a detailed description of all possible AC-3 Controls a CODEC function block can incorporate. For each Control, the layout of the parameter block together with the appropriate Control Selector are listed.

10.6.4.3.1 Valid ChannelConfig Values (informative)

No Mode[I] specified for an AC-3 decoder may set bit D14 in its ChannelConfig field. Logical channels produced by an AC-3 decoder follow a specific naming convention and should not be changed. Non-conventional channel ordering is allowed and bit D15 shall be set for this purpose. There are only certain bits in the ChannelConfig fields (conventional ordering) corresponding to Mode[I] that are valid when describing the output of an AC-3 decoder.

An implementation of an AC-3 decoder must support some subset of the following channel configurations:

- 0004_{16}: C
- 0003_{16}: L, R
- 0007_{16}: L, C, R
- 0103_{16}: L, R, S
- 0107_{16}: L, C, R, S
- 0033_{16}: L, R, LS, RS
- 0037_{16}: L, C, R, LS, RS
- $000C_{16}$: C, LFE
- $000B_{16}$: L, R, LFE
- $000F_{16}$: L, C, R, LFE

— $010B_{16}$: L, R, S, LFE

— $010F_{16}$: L, C, R, S, LFE

— $003B_{16}$: L, R, LS, RS, LFE

— $003F_{16}$: L, C, R, LS, RS, LFE

— $013F_{16}$: L, C, R, LS, RS, S, LFE

10.6.4.3.2 Mode Control

The Mode Control is used to change the compression mode of the AC-3 decoder. A mode control can only support the CURRENT attribute. The valid range for the CURRENT attribute is described through the **ComprFeatures** field of the AC-3 format specific descriptor. Bits D3…0 describe which compression modes the AC-3 decoder supports. Valid values are:

— 0: RF mode

— 1: Line mode

— 2: Custom0 mode

— 3: Custom1 mode

Also the value FF_{16} indicates invalidity and shall be not used for a control command.

If the Mode Control command specifies an unsupported mode, the response frame must indicate NOT IMPLEMENTED. The current setting can be queried using an audio command addressing the AC-3 mode control.

	msb							lsb
Operand[4]	Control Selector							
Operand[5]	Length of control parameters (1)							
Operand[5+1]	Mode							

Figure 10.113 – AC-3 CODEC Control Parameter Block

Operand[4]	MODE_CONTROL
Operand[5]	1

Figure 10.114 – Values for the fields in the first form of the …Control Parameters

Table 10.18 – Values of the Mode parameter

Value	Description
0	RF mode
1	Line Mode
2	Custom0 mode
3	Cutom1 mode

10.6.4.3.3 Dynamic Range Control

The Dynamic Range Control (DRC) is used to enable or disable the Dynamic Range Control functionality of the decoder. The Dynamic Range Control can have only the current setting attribute (CURRENT). The position of the Dynamic Range Control switch can be either TRUE (70_{16}) or FALSE (60_{16}). Also the value FF_{16} indicates invalidity and shall not be used for a control command. TRUE means that the AC-3 decoder is using the Dynamic Range control words (possibly with additional scaling) contained in the AC-3 bit stream to control the audio dynamic range. FALSE means the control words are being ignored and the original signal dynamic range is being reproduced. The current setting of the Control can be queried using an audio command addressing the AC-3 Dynamic Range Control.

	msb							lsb
Operand[4]	Control Selector							
Operand[5]	Length of control parameters (1)							
Operand[5+1]	Enable_Dynamic_Range							

Figure 10.115 – AC-3 Processing Control Parameter Block

Operand[4]	AC3_DYNAMIC_RANGE_CONTROL
Operand[5]	1

Figure 10.116 – Values for the fields in the first form of the …Control Parameters

10.6.4.3.4 Scaling Control

The Scaling Control is used to manipulate the single scaling coefficient used by AC-3 decoders that implement a common boost/cut scaling value for Dynamic Range Control. (D5..4 = '10' in the **AC3Features** field of the AC-3 format-specific descriptor.) If this Control is addressed on a non-'10' decoder, the response frame must return NOT IMPLEMENTED.

The Scaling Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The valid range for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes is from zero (00_{16}) to 254/256 (FE_{16}). Also the value FF_{16} indicates invalidity and shall not be used for a control command. The Scaling Control honors the request to the best of its abilities. It may round the **Scale** attribute value to its closest available setting. It will report this rounded setting when queried using an audio command addressing the scaling control.

	msb							lsb
Operand[4]	Control Selector							
Operand[5]	Length of control parameters (1)							
Operand[5+1]	Scale							

Figure 10.117 – AC-3 Processing Control Parameter Block

Operand[4]	AC3_SCALE_CONTROL
Operand[5]	1

Figure 10.118 – Values for the fields of the …Control Parameters

10.6.4.3.5 High/Low Scaling Control

The High/Low Scaling Control is used to manipulate the two scaling coefficients used by AC-3 decoders that implement an independent boost and cut scaling value for Dynamic Range Control. (D5..4 = '11' in the **AC3Features** field of the AC-3 format-specific descriptor.) If this Control is addressed on a non-'11' decoder, the response frame must indicate NOT IMPLEMENTED.

The High/Low Scaling Control can support all possible Control attributes (CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA). The valid range for the CURRENT, MINIMUM, MAXIMUM, DEFAULT RESOLUTION and DELTA attributes is from zero (00_{16}) to 254/256 (FE_{16}). Also the value FF_{16} indicates invalidity and shall not be used for a control command. The High/Low Scaling Control honors the request to the best of its abilities. It may round the **LowScale** and **HighScale** attribute values to their closest available settings. It will report these rounded settings when queried by an audio command. The **LowScale** value is used by the AC-3 decoder to scale the Dynamic Range control words which apply a gain increase (for low sound levels). The **HighScale** value is used by the AC-3 decoder to scale the Dynamic Range control words which apply a gain reduction (for high level sounds).

	msb							lsb
Operand[4]	Control Selector							
Operand[5]	Length of control parameters (2)							
Operand[5+1]	HighScale							
Operand[5+2]	LowScale							

Figure 10.119 – AC-3 High/Low Scaling Control Parameter Block

Operand[4]	AC3_HL_SCALING_CONTROL
Operand[5]	2

Figure 10.120 – Values for the fields High/Low ScalingControl Parameters

10.6.4.3.6 Dual Mono Control

The Dual Mono Mode Control is used to select which of the two available channels in an AC-3 dual mono bitstream is actually retrieved and reproduced over the Left and Right output channels.

The Dual Mono Control can have only the current setting attribute (CURRENT). The current setting of the Control can be queried using a Get AC-3 Control request. A decoder does not have to allow setting of this control's CURRENT value, but it must at the very least return AC3_DUAL_MONO_STEREO (indicating Ch. 1 & 2 as stereo) when this control's current setting is queried. If any Controller Device attempts to set this control's CURRENT value to a Dual Mono mode that is not indicated as supported by bits D7..6 in the AC3Features field of the AC-3 format-specific descriptor, the response frame must indicate NOT IMPLEMENTED.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	Dual_Mono_Mode

Figure 10.121 – Dual Mono Control Parameter Block

Operand[4]	AC3_DUAL_MONO_CONTROL
Operand[5]	1

Figure 10.122 – Values for Dual Mono Control Parameter Block

AC3_DUAL_MONO_STEREO	Ch. 1 & 2 as stereo
AC3_DUAL_MONO_CH1	Ch. 1 as mono
AC3_DUAL_MONO_CH2	Ch. 2 as mono
AC3_DUAL_MONO_MIXED	Ch. 1 & 2 mixed as mono

Figure 10.123 – Values of the Dual_Mono_Mode parameter

The FUNCTION BLOCK command with a *control_selector* of AC_3_DUAL_MONO_CONTROL may be used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current setting of the control_attribute when the STABLE response frame is returned.

The *operand[7]* in the control command shall not be set to FF_{16}.

10.6.4.3.7 Dolby Surround Output Control

This control is used to enable or disable Dolby Surround decoding of the output PCM audio when the input Dolby Digital bitstream contains only two channels (Left and Right).

This control will have no effect when the input bitstream contains channels other than (or in addition to) Left and Right. The Dolby Surround Output Control can have only the current setting attribute (CURRENT). The position of the Dolby Surround Output Control switch can be either TRUE (70_{16}) or FALSE (60_{16}). Also the value FF_{16} indicates invalidity and shall not be used for a control command. TRUE means that the Left and Right PCM channels being output from the AC-3 decoder will be processed by a Dolby Surround decoding algorithm to retrieve up to two more channels of audio (C and/or S). FALSE means the Left and Right PCM audio will not have Dolby Surround decode processing applied to it, meaning that if the 2-channel audio data are Dolby Pro-logic encoded, they will exit the function block still Pro-logic encoded. The current setting of the Control can be queried using an audio command addressing the AC-3 Dolby Surround Output Control.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	Enable_Dolby_Surround

Figure 10.124 – Dolby Surround Control Parameter Block

Operand[4]	AC3_DOLBY_SURROUND_CONTROL
Operand[5]	1

Figure 10.125 – Values for the fields in the Dolby Surround Output Control Parameters

10.6.4.3.8 Frame Error Status Control

This control is used for querying the error state of an AC-3 decoder for the AC-3 frame currently being processed by the decoder.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_FRAME_ERROR_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current error state of the AC-3 decoder when the STABLE response frame is returned

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	Frame_Error_Status

Figure 10.126 – Frame Error Status Control Parameter Block

Operand[4]	AC3_FRAME_ERROR_STATUS_CONTROL
Operand[5]	1

Figure 10.127 – Values for Frame Error Status Control Parameter Block

AC3_NOFRAME_ERRORS	No AC-3 frame errors occurred
AC3_INVALID_FRAME_SYNC	The decoder is not receiving a valid AC-3 bitstream.
AC3_INVALID_SAMPLE_RATE	The sample rate embedded in the bitstream is unsupported by the decoder
AC3_INVALID_DATA_RATE	The data rate embedded in the bitstream is unsupported by the decoder.
AC3_UNSUP_BSID	The decoder does not support the version number of the current bitstream.

Figure 10.128 – Values of the Frame_Error_Status parameter

10.6.4.3.9 Sample Rate Status Control

This control is used for querying an AC-3 decoder for the embedded sample rate of the AC-3 frame currently being processed by the decoder.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_SAMPLE_RATE_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current embedded sample rate of the AC-3 frame when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	Sample_Rate_Status

Figure 10.129 – Sample Rate Status Control Parameter Block

Operand[4]	AC3_SAMPLE_RATE_STATUS_CONTROL
Operand[5]	1

Figure 10.130 – Values for Sample Rate Status Control Parameter Block

AC3_48KHZ	The original PCM sample rate of the current AC-3 frame is 48kHz.
AC3_441KHZ	The original PCM sample rate of the current AC-3 frame is 44.1kHz.
AC3_32KHZ	The original PCM sample rate of the current AC-3 frame is 32kHz.

Figure 10.131 – Values of the Sample_Rate_Status parameter

10.6.4.3.10 Data Rate Status Control

This control is used for querying an AC-3 decoder for the embedded data rate of the AC-3 frame currently being processed by the decoder.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_DATA_RATE_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[6]* is set to FF_{16} when the status command is issued and is updated to the current embedded data rate of the AC-3 frame when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	Data_Rate_Status

Figure 10.132 – Data Rate Status Control Parameter Block

Operand[4]	AC3_DATA_RATE_STATUS_CONTROL
Operand[5]	1

Figure 10.133 – Values for Data Rate Status Control Parameter Block

Table 10.19 – Values of the Data_Rate_Status parameter

contents	description
AC3_DATARATE_32KBS	AC-3 data requires 32 kbits per second for real-time transmission.
AC3_DATARATE_40KBS	AC-3 data requires 40 kbits per second for real-time transmission.
AC3_DATARATE_48KBS	AC-3 data requires 48 kbits per second for real-time transmission.
AC3_DATARATE_56KBS	AC-3 data requires 56 kbits per second for real-time transmission.
AC3_DATARATE_64KBS	AC-3 data requires 64 kbits per second for real-time transmission.
AC3_DATARATE_80KBS	AC-3 data requires 80 kbits per second for real-time transmission.
AC3_DATARATE_96KBS	AC-3 data requires 96 kbits per second for real-time transmission.
AC3_DATARATE_112KBS	AC-3 data requires 112 kbits per second for real time transmission.
AC3_DATARATE_128KBS	AC-3 data requires 128 kbits per second for real- time transmission.
AC3_DATARATE_160KBS	AC-3 data requires 160 kbits per second for real-time transmission.
AC3_DATARATE_192KBS	AC-3 data requires 192 kbits per second for real-time transmission.
AC3_DATARATE_224KBS	AC-3 data requires 224 kbits per second for real-time transmission.
AC3_DATARATE_256KBS	AC-3 data requires 256 kbits per second for real-time transmission.
AC3_DATARATE_320KBS	AC-3 data requires 320 kbits per second for real-time transmission.
AC3_DATARATE_384KBS	AC-3 data requires 384 kbits per second for real-time transmission.
AC3_DATARATE_448KBS	AC-3 data requires 448 kbits per second for real-time transmission.
AC3_DATARATE_512KBS	AC-3 data requires 512 kbits per second for real-time transmission.
AC3_DATARATE_576KBS	AC-3 data requires 576 kbits per second for real-time transmission.
AC3_DATARATE_640KBS	AC-3 data requires 640 kbits per second for real-time transmission.

10.6.4.3.11 LFE On Status Control

This control is used for querying an AC-3 decoder to discover if the AC-3 frame currently being processed contains Low Frequency Enhancement information. The value of this control can be only be TRUE (70_{16}) or FALSE (60_{16}). TRUE indicates the presence of LFE information, FALSE indicates its absence.

The FUNCTION BLOCK command with a *control_selector* of AC3_LFE_ON_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	LFE_On

Figure 10.134 – LFE On Status Control Parameter Block

Operand[4]	AC3_LFE_ON_STATUS_CONTROL
Operand[5]	1

Figure 10.135 – Values for the fields in the LFE On Status Control Parameters

10.6.4.3.12 Audio Coding Mode Status Control

This control is used for querying an AC-3 decoder for the embedded audio coding mode of the AC-3 frame currently being processed by the decoder.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_MOD_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current embedded audio coding mode of the AC-3 frame when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	acmod_Status

Figure 10.136 – Audio Coding Mode Status Control Parameter Block

Operand[4]	AC3_MOD_STATUS_CONTROL
Operand[5]	1

Figure 10.137 – Values for Audio Coding Mode Status Control Parameter Block

AC3_IN_MODE11	AC-3 data is in 1+1 (dual mono) mode.
AC3_IN_MODE10	AC-3 data is in 1/0 (C) mode.
AC3_IN_MODE20	AC-3 data is in 2/0 (L,R) mode.
AC3_IN_MODE30	AC-3 data is in 3/0 (L,C,R) mode.
AC3_IN_MODE21	AC-3 data is in 2/1 (L,R,S) mode.
AC3_IN_MODE31	AC-3 data is in 3/1 (L,C,R,S) mode.
AC3_IN_MODE22	AC-3 data is in 2/2 (L,R,LS,RS) mode.
AC3_IN_MODE32	AC-3 data is in 3/2 (L,C,R,LS,RS) mode.

Figure 10.138 – Values of the acmod_Status parameter

10.6.4.3.13 Bitstream ID status Control

This control is used for querying an AC-3 decoder for the embedded bitstream ID of the AC-3 frame currently being processed by the decoder.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_BSID_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current embedded bitstream ID of the AC-3 frame when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	bsid_Status

Figure 10.139 – Bitstream ID Status Control Parameter Block

Operand[4]	AC3_BSID_STATUS_CONTROL
Operand[5]	1

Figure 10.140 – Values for Bitstream ID Status Control Parameter Block

BSID values can range from 0 to 31. Standard AC-3 decoders must be capable of processing at least BSIDs 0 to 8 (meaning they are capable of interpreting the version 8 of the AC-3 bit-stream syntax and subsets).

10.6.4.3.14 Bitstream Mode Status Control

This control is used for querying an AC-3 decoder for the embedded bitstream mode of the AC-3 frame currently being processed by the decoder. This information is not used by the decoder itself, but is merely used to indicate information about the audio content.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_BSMOD_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current embedded bitstream mode of the AC-3 frame when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	bsmod_Status

Figure 10.141 – Bitstream Mode Status Control Parameter Block

Operand[4]	AC3_BSMOD_STATUS_CONTROL
Operand[5]	1

Figure 10.142 – Values for Bitstream Mode Status Control Parameter Block

AC3_BSMOD_MAIN_AUDIO	1
AC3_BSMOD_NO_DIALOG	2
AC3_BSMOD_VISUALLY_IMPAIRED	3
AC3_BSMOD_HEARING_IMPAIRED	4
AC3_BSMOD_DIALOG_ONLY	5
AC3_BSMOD_COMMENTARY	6
AC3_BSMOD_EMERGENCY	7
AC3_BSMOD_VOICE_OVER	8
AC3_BSMOD_KARAOKE	9

Figure 10.143 – Values of the bsmod_Status parameter

10.6.4.3.15 Center Mix Level Status Control

This control is used for querying an AC-3 decoder for the embedded center mix level of the AC-3 frame currently being processed by the decoder. When three front channels are in use, this code indicates the nominal down mix level of the center channel with respect to the left and right channels.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_CMIXLEV_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current embedded center mix level of the AC-3 frame when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	cmixlev_Status

Figure 10.144 – Center Mix Level Control Parameter Block

Operand[4]	AC3_CMIXLEV_STATUS_CONTROL
Operand[5]	1

Figure 10.145 – Values for Center Mix Level Status Control Parameter Block

AC3_CMIXLEV_3DB	0.707 (-3.0 dB)
AC3_CMIXLEV_45DB	0.596 (-4.5 dB)
AC3_CMIXLEV_6DB	0.500 (-6.0 dB)
AC3_CMIXLEV_NOCENTER	no center channel present

Figure 10.146 – Values of the cmixlev_Status parameter

10.6.4.3.16 Surround Mix Level Status Control

This control is used for querying an AC-3 decoder for the embedded surround mix level of the AC-3 frame currently being processed by the decoder. When surround channels are in use, this code indicates the nominal down mix level of the surround channels with respect to the front left and right channels.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_SMIXLEV_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current embedded surround mix level of the AC-3 frame when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	smixlev_Status

Figure 10.147 – Surrround Mix Level Status Control Parameter Block

Operand[4]	AC3_SMIXLEV_STATUS_CONTROL
Operand[5]	1

Figure 10.148 – Values for Surrround Mix Level Status Control Parameter Block

AC3_SMIXLEV_3DB	0.707 (-3.0 dB)
AC3_SMIXLEV_6DB	0.500 (-6.0 dB)
AC3_SMIXLEV_INFDB	0.000 (-infinity dB)
AC3_SMIXLEV_NOSURROUND	no surround channel(s) present

Figure 10.149 – Values of the smixlev_Status parameter

10.6.4.3.17 Dolby Surround Status Control

This control is used for querying an AC-3 decoder for the embedded Dolby Surround Encoding indicator of the AC-3 frame currently being processed by the decoder. When a two channel incoming bitstream is present, this code indicates whether the incoming program material has been encoded in Dolby Surround. This control's value is not valid when anything other than an L, R input signal is present.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_DSURR_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current embedded Dolby Surround Encoding indicator of the AC-3 frame when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	dsurr_Status

Figure 10.150 – Dolby Surround Status Control Parameter Block

Operand[4]	AC3_DSURR_STATUS_CONTROL
Operand[5]	1

Figure 10.151 – Values for Dolby Surround Status Control Parameter Block

AC3_DSURR_NA	Dolby Surround not indicated
AC3_DSURR_ENC	Dolby Surround encoding present
AC3_DSURR_NOENC	Dolby Surround encoding absent

Figure 10.152 – Values of the dsurr_Status parameter

10.6.4.3.18 Copyright Status Control

This control is used for querying an AC-3 decoder for the embedded copyright indicator of the AC-3 frame currently being processed by the decoder. This control's value can be either TRUE or FALSE. TRUE indicates that the bitstream is protected by copyright. FALSE indicates that it is not.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_CPYRT_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current embedded copyright indicator of the AC-3 frame when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	cpyrt_Status

Figure 10.153 – Copyright Status Control Parameter Block

Operand[4]	AC3_CPYRT_STATUS_CONTROL
Operand[5]	1

Figure 10.154 – Values for Copyright Status Control Parameter Block

10.6.4.3.19 Original Bitstream Status Control

This control is used for querying an AC-3 decoder for the embedded original bitstream indicator of the AC-3 frame currently being processed by the decoder. This control's value can be either TRUE (70_{16}) or FALSE (60_{16}). TRUE indicates that the bitstream is an original. FALSE indicates that this bitstream is a copy of another bitstream.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_ORGNL_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current embedded bitstream indicator of the AC-3 frame when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	orgnl_Status

Figure 10.155 – Original Bitstream Status Control Parameter Block

Operand[4]	AC3_ORGNL_STATUS_CONTROL
Operand[5]	1

Figure 10.156 – Values for Original Bitstream Status Control Parameter Block

10.6.4.3.20 Dialog Normalization Status Control

This control is used for querying an AC-3 decoder for the embedded dialog normalization value of the AC-3 frame currently being processed by the decoder. This control indicates how far the average dialog level is below digital 100%..

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_DIALNORM_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current value of dialnorm_Status when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	dialnorm_Status

Figure 10.157 – Dialnorm Status Control Parameter Block

Operand[4]	AC3_DIALNORM_STATUS_CONTROL
Operand[5]	1

Figure 10.158 – Values for Dialog Nomalization Status Control Parameter Block

Valid values for dialnorm_Status range from 1-31, indicating -1 dB to -31 dB below digital 100% respectively.

10.6.4.3.21 Dialog Normalization 2 Status Control

This control is used for querying an AC-3 decoder for the embedded dialog normalization value of channel 2 of a dual-mono input stream for the AC-3 frame currently being processed by the decoder. Valid values for dialnorm2_Status behave exactly as values of dialnorm_Status, except that dialnorm2_Status values apply only when the input bitstream is in dual-mono mode and only to the second of the two independent mono channels.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_DIALNORM2_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current value of dialnorm2_Status when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	dialnorm2_Status

Figure 10.159 – Dialnorm Status Control Parameter Block

Operand[4]	AC3_DIALNORM2_STATUS_CONTROL
Operand[5]	1

Figure 10.160 – Values for Dialog Nomalization Status Control Parameter Block

10.6.4.3.22 Mix Level Status Control

This control is used for querying an AC-3 decoder for the embedded recording studio mixing level for the AC-3 frame currently being processed by the decoder. This control indicates the absolute acoustic SPL of the content during the final audio mixing session.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_MIXLEV_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current embedded recording studio mixing level of the AC-3 frame when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	mixlev_Status

Figure 10.161 – Mix Level Status Control Parameter Block

Operand[4]	AC3_MIXLEV_STATUS_CONTROL
Operand[5]	1

Figure 10.162 – Values for Mix Level Status Control Parameter Block

Valid values for mixlev_Status range from 0 to 31, representing 80 to 111 dB

SPL in single decibel increments.

10.6.4.3.23 Mix Level 2 Status Control

This control is used for querying an AC-3 decoder for the embedded Mixer Level value of channel 2 of a dual-mono input stream for the AC-3 frame currently being processed by the decoder. Valid values for mixlev2_Status behave exactly as values of mixlev2_Status, except that mixlev2_Status values apply only when the input bitstream is in dual-mono mode and only to the second of the two independent mono channels.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_MIXLEV2_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current values of mixlev2_Status when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	mixlev2_Status

Figure 10.163 – Mix Level 2 Status Control Parameter Block

Operand[4]	AC3_MIXLEV2_STATUS_CONTROL
Operand[5]	1

Figure 10.164 – Values for Mix Level 2 Status Control Parameter Block

10.6.4.3.24 Room Type Status Control

This control is used for querying an AC-3 decoder for the embedded room type value. This control indicates the room typ and calibration of the mixing room used for the final audio session.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_ROOMTYPE_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current embedded room type value of the AC-3 frame when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	roomtype_Status

Figure 10.165 – Room Type Status Control Parameter Block

Operand[4]	AC3_ROOMTYPE_STATUS_CONTROL
Operand[5]	1

Figure 10.166 – Values for Room Type Status Control Parameter Block

AC3_ROOMTYPE_NOT_INDICATED	the bitstream does not indicate the room type
AC3_ROOMTYPE_LARGEX	Large room, X curve monitor
AC3_ROOMTYPE_SMALLFLAT	Small room, flat montor

Figure 10.167 – Values of the roomtype_Status parameter

10.6.4.3.25 Room Type 2 Status Control

This control is used for querying an AC-3 decoder for the embedded Room Type value of channel 2 of a dual-mono input stream for the AC-3 frame currently being processed by the decoder. Valid values for roomtype2_Status behave exactly as values of roomtype2_Status, except that roomtype2_Status values

apply only when the input bitstream is in dual-mono mode and only to the second of the two independent mono channels.

This control does not accept commands with ctype = CONTROL or ctype = NOTIFY. The value of this control is set by the function block based upon data in the incoming bitstream.

The FUNCTION BLOCK command with a *control_selector* of AC3_ROOMTYPE2_STATUS_CONTROL is used as status command. This command has a *ctype* value of STATUS. In this case, *operand[7]* is set to FF_{16} when the status command is issued and is updated to the current value of roomtype2_Status when the STABLE response frame is returned.

Operand[4]	Control Selector
Operand[5]	Length of control parameters (1)
Operand[5+1]	roomtype2_Status

Figure 10.168 – Room Type 2 Status Control Parameter Block

Operand[4]	AC3_ROOMTYPE2_STATUS_CONTROL
Operand[5]	1

Figure 10.169 – Values for Room Type 2 Status Control Parameter Block

11. Audio Subunit Commands

This clause defines commands that are applicable to an audio subunit.

Table 11.1 – Audio subunit commands

Opcode	value	Supported ctype			Comments
		C	S	N	
CHANGE CONFIGURATION	$C0_{16}$	R	R	O	Reconfigure the audio subunit

In the preceding table, a dash in the support level column indicates that the command is not defined for the *ctype* value, CONTROL, STATUS or NOTIFY, indicated. The specific command formats and corresponding response frame formats are described for each of the audio subunit commands in the sub-clauses that follow.

11.1 CHANGE CONFIGURATION command

The CHANGE CONFIGURATION command is used to reconfigure the audio subunit. Only existing configurations can be included in operand[0]-[1] of the command. The *configuration_ID* is specified in each *configuration_dependent_information*. The *configuration_ID* must be unique for a given audio subunit.

	msb							lsb
	0000				ctype			
	subunit_type					subunit_ID		
Opcode	CHANGE CONFIGURATION ($C0_{16}$)							
Operand[0]	configuration_ID (MSB)							
Operand[1]	(LSB)							

Figure 11.1 – CHANGE CONFIGURATION command

When the CHANGE CONFIGURATION command is issued with a *ctype* value of CONTROL, the *configuration_ID* field specifies the desired value of a *configuration_ID*.

The CHANGE CONFIGURATION command with a *ctype* value of STATUS may be used to determine the currrent configuration state of the audio subunit. In this case, operand[0]-[1] is set to $FFFF_{16}$ when the status command is issued and is updated to the current configuration_ID when the STABLE response frame is returned.

The CHANGE CONFIGURATION command may also be used as notify command. The notify command has the same syntax as the status command. A notification shall be returned by the target to the controller that issued the notify command in case the configuration state of the addressed audio subunit changes. The notify response has the same format as the status response frame.

Table 11.2 – configuration_ID

configuration_ID	Meaning
0000_{16}	reserved
0001_{16} - $FFFE_{16}$	audio subunit dependent
$FFFF_{16}$	invalid

Annexes

A. Encoding Tables

A.1 Function Block Type encoding

Table A.1 – Function Block Type encoding

Function Block Type	Value
SELECTOR	80_{16}
FEATURE	81_{16}
PROCESSING	82_{16}
CODEC	83_{16}
Reserved for future use	84_{16}-$8F_{16}$

A.2 Processing Type encoding

Table A.2 – Processing type encoding

Processing Type	Value
MIXER	01_{16}
GENERIC	02_{16}
UP_DOWN	03_{16}
DOLBY_PRO_LOGIC	04_{16}
3-D STEREO EXTENDER	05_{16}
REVERBERATION	06_{16}
CHORUS	07_{16}
DYNAMIC_RANGE_COMPRESSION	08_{16}
reserved	09_{16}-FF_{16}

Table A.3 – CODEC type encoding

CODEC Type	Value
AC3_DECODER	01_{16}
MPEG_DECODER	02_{16}
DTS_DECODER	03_{16}
reserved	04_{16}-FF_{16}

A.3 Control Selector Encoding

Table A.4 – Control Selector Encoding

function block type	sub-type	control_selector	Value
selector		SELECTOR_CONTROL	01_{16}
feature		MUTE_CONTROL	01_{16}
		VOLUME_CONTROL	02_{16}
		LR_BALANCE_CONTROL	03_{16}
		FR_BALANCE_CONTROL	04_{16}
		BASS_CONTROL	05_{16}
		MID_CONTROL	06_{16}
		TREBLE_CONTROL	07_{16}
		GEQ_CONTROL	08_{16}
		AGC_CONTROL	09_{16}
		DELAY_CONTROL	$0A_{16}$
		BASSBOOST_CONTROL	$0B_{16}$
		LOUDNESS_CONTROL	$0C_{16}$
processing		ENABLE_CONTROL	01_{16}
		MODE_CONTROL	02_{16}
	mixer	MIXER_CONTROL	03_{16}
	3D-stereo extender	SPACIOUSNESS_CONTROL	03_{16}
	reverberation	REVERBTYPE_CONTROL	03_{16}
		REVERBLEVEL_CONTROL	04_{16}
		REVERBTIME_CONTROL	05_{16}
		REVERBEARLYTIME_CONTROL	06_{16}
		REVERBDELAY_CONTROL	07_{16}
	chorus	CHORUSRATE_CONTROL	03_{16}
		CHORUSDEPTH_CONTROL	04_{16}
	dynamic range compressor	COMPRESSION_RATIO_CONTROL	03_{16}
		MAXAMPL_CONTROL	04_{16}
		THRESHOLD_CONTROL	05_{16}
		ATTACKTIME_CONTROL	06_{16}
		RELEASETIME_CONTROL	07_{16}
	generic	GUID_CONTROL	03_{16}
CODEC		ENABLE_CONTROL	01_{16}
		MODE_CONTROL	02_{16}

function block type	sub-type	control_selector	Value
	MPEG decoder	MPEG_DUAL_CHANNEL_CONTROL	03_{16}
		MPEG_SECOND_STEREO_CONTROL	04_{16}
		MPEG_MULTILINGUAL_CONTROL	05_{16}
		MPEG_DYN_RANGE_CONTROL	06_{16}
		MPEG_SCALE_CONTROL	07_{16}
		HL_SCALING_CONTROL	08_{16}
	AC3 decoder	AC3_DYNAMIC_RANGE_CONTROL	03_{16}
		AC3_SCALE_CONTROL	04_{16}
		AC3_HL_SCALING_CONTROL	05_{16}
		AC3_DUAL_MONO_CONTROL	06_{16}
		AC3_DOLBY_SURROUND_CONTROL	07_{16}
		AC3_FRAME_ERROR_STATUS_CONTROL	08_{16}
		AC3_SAMPLE_RATE_STATUS_CONTROL	09_{16}
		AC3_DATA_RATE_STATUS_CONTROL	$0A_{16}$
		AC3_LFE_ON_STATUS_CONTROL	$0B_{16}$
		AC3_MOD_STATUS_CONTROL	$0C_{16}$
		AC3_BSID_STATUS_CONTROL	$0D_{16}$
		AC3_BSMOD_STATUS_CONTROL	$0E_{16}$
		AC3_CMIXLEV_STATUS_CONTROL	$0F_{16}$
		AC3_SMIXLEV_STATUS_CONTROL	10_{16}
		AC3_DSUR_STATUS_CONTROL	11_{16}
		AC3_CPYRT_STATUS_CONTROL	12_{16}
		AC3_ORGNL_STATUS_CONTROL	13_{16}
		AC3_DIALNORM_STATUS_CONTROL	14_{16}
		AC3_DIALNORM2_STATUS_CONTROL	15_{16}
		AC3_MIXLEV_STATUS_CONTROL	16_{16}
		AC3_MIXLEV2_STATUS_CONTROL	17_{16}
		AC3_ROOMTYPE_STATUS_CONTROL	18_{16}
		AC3_ROOMTYPE2_STATUS_CONTROL	19_{16}

All other values reserved for future use.